

Melissa

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
4.1	misc functions . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Function Documentation . . . . .	7
4.1.2.1	clear_bit() . . . . .	7
4.1.2.2	melissa_bind() . . . . .	8
4.1.2.3	melissa_calloc() . . . . .	8
4.1.2.4	melissa_connect() . . . . .	8
4.1.2.5	melissa_free() . . . . .	9
4.1.2.6	melissa_get_node_name() . . . . .	9
4.1.2.7	melissa_get_time() . . . . .	9
4.1.2.8	melissa_logo() . . . . .	9
4.1.2.9	melissa_malloc() . . . . .	9
4.1.2.10	set_bit() . . . . .	10
4.1.2.11	test_bit() . . . . .	10
4.2	internal API . . . . .	11

4.2.1	Detailed Description	11
4.2.2	Function Documentation	11
4.2.2.1	compute_stats()	11
4.2.2.2	finalize_stats()	11
4.3	Melissa fields	12
4.4	Melissa data	13
4.4.1	Detailed Description	13
4.4.2	Function Documentation	13
4.4.2.1	melissa_free_data()	13
4.4.2.2	melissa_init_data()	13
4.5	input, output and checkpoint functions	14
4.5.1	Detailed Description	14
4.5.2	Function Documentation	14
4.5.2.1	read_client_data()	14
4.5.2.2	read_ensight()	14
4.5.2.3	read_saved_stats()	15
4.5.2.4	read_simu_states()	15
4.5.2.5	save_simu_states()	16
4.5.2.6	save_stats()	16
4.5.2.7	write_client_data()	16
4.5.2.8	write_stats_bin()	17
4.5.2.9	write_stats_ensight()	17
4.5.2.10	write_stats_txt()	17
4.6	Get options from command line	19
4.6.1	Detailed Description	19
4.6.2	Function Documentation	19
4.6.2.1	melissa_check_options()	19
4.6.2.2	melissa_get_options()	19
4.6.2.3	melissa_print_options()	20
4.6.2.4	melissa_read_options()	20
4.6.2.5	melissa_write_options()	20
4.7	API	21
4.7.1	Detailed Description	21
4.7.2	Function Documentation	21
4.7.2.1	melissa_finalize()	21
4.7.2.2	melissa_init()	21
4.7.2.3	melissa_init_no_mpi()	22
4.7.2.4	melissa_send()	22
4.7.2.5	melissa_send_no_mpi()	22

<b>5</b>	<b>Class Documentation</b>	<b>25</b>
5.1	comm_data_s Struct Reference . . . . .	25
5.1.1	Detailed Description . . . . .	25
5.1.2	Member Data Documentation . . . . .	25
5.1.2.1	client_comm_size . . . . .	25
5.1.2.2	comm_size . . . . .	25
5.1.2.3	rank . . . . .	26
5.1.2.4	rcounts . . . . .	26
5.1.2.5	rdispls . . . . .	26
5.2	covariance_s Struct Reference . . . . .	26
5.2.1	Detailed Description . . . . .	26
5.2.2	Member Data Documentation . . . . .	26
5.2.2.1	covariance . . . . .	27
5.2.2.2	increment . . . . .	27
5.2.2.3	mean1 . . . . .	27
5.2.2.4	mean2 . . . . .	27
5.3	mean_s Struct Reference . . . . .	27
5.3.1	Detailed Description . . . . .	27
5.3.2	Member Data Documentation . . . . .	27
5.3.2.1	increment . . . . .	28
5.3.2.2	mean . . . . .	28
5.4	melissa_data_s Struct Reference . . . . .	28
5.4.1	Detailed Description . . . . .	28
5.4.2	Member Data Documentation . . . . .	28
5.4.2.1	free_sobol . . . . .	29
5.4.2.2	increment_sobol . . . . .	29
5.4.2.3	init_sobol . . . . .	29
5.4.2.4	is_valid . . . . .	29
5.4.2.5	means . . . . .	29
5.4.2.6	min_max . . . . .	29

5.4.2.7	<code>nb_simu</code>	29
5.4.2.8	<code>options</code>	29
5.4.2.9	<code>quantiles</code>	30
5.4.2.10	<code>read_sobol</code>	30
5.4.2.11	<code>save_sobol</code>	30
5.4.2.12	<code>sobol_indices</code>	30
5.4.2.13	<code>step_simu</code>	30
5.4.2.14	<code>thresholds</code>	30
5.4.2.15	<code>variances</code>	30
5.4.2.16	<code>vect_size</code>	31
5.5	<code>melissa_field_s</code> Struct Reference	31
5.5.1	Detailed Description	31
5.5.2	Member Data Documentation	31
5.5.2.1	<code>name</code>	31
5.5.2.2	<code>stats_data</code>	31
5.6	<code>melissa_options_s</code> Struct Reference	32
5.6.1	Detailed Description	32
5.6.2	Member Data Documentation	32
5.6.2.1	<code>global_vect_size</code>	32
5.6.2.2	<code>launcher_name</code>	32
5.6.2.3	<code>mean_op</code>	33
5.6.2.4	<code>min_and_max_op</code>	33
5.6.2.5	<code>nb_fields</code>	33
5.6.2.6	<code>nb_parameters</code>	33
5.6.2.7	<code>nb_simu</code>	33
5.6.2.8	<code>nb_time_steps</code>	33
5.6.2.9	<code>quantile_op</code>	33
5.6.2.10	<code>restart</code>	33
5.6.2.11	<code>restart_dir</code>	34
5.6.2.12	<code>sampling_size</code>	34

5.6.2.13	<a href="#">sobol_op</a>	34
5.6.2.14	<a href="#">sobol_order</a>	34
5.6.2.15	<a href="#">threshold</a>	34
5.6.2.16	<a href="#">threshold_op</a>	34
5.6.2.17	<a href="#">variance_op</a>	34
5.7	<a href="#">melissa_simulation_s Struct Reference</a>	35
5.8	<a href="#">min_max_s Struct Reference</a>	35
5.8.1	<a href="#">Detailed Description</a>	35
5.8.2	<a href="#">Member Data Documentation</a>	35
5.8.2.1	<a href="#">is_init</a>	35
5.8.2.2	<a href="#">max</a>	35
5.8.2.3	<a href="#">min</a>	36
5.9	<a href="#">pull_data_s Struct Reference</a>	36
5.9.1	<a href="#">Detailed Description</a>	36
5.9.2	<a href="#">Member Data Documentation</a>	36
5.9.2.1	<a href="#">buff_size</a>	36
5.9.2.2	<a href="#">local_nb_messages</a>	36
5.9.2.3	<a href="#">message_sizes</a>	37
5.9.2.4	<a href="#">pull_rank</a>	37
5.9.2.5	<a href="#">push_rank</a>	37
5.9.2.6	<a href="#">total_nb_messages</a>	37
5.10	<a href="#">quantile_s Struct Reference</a>	37
5.10.1	<a href="#">Detailed Description</a>	37
5.10.2	<a href="#">Member Data Documentation</a>	38
5.10.2.1	<a href="#">alpha</a>	38
5.10.2.2	<a href="#">increment</a>	38
5.10.2.3	<a href="#">quantile</a>	38
5.11	<a href="#">sobol_array_s Struct Reference</a>	38
5.11.1	<a href="#">Detailed Description</a>	38
5.11.2	<a href="#">Member Data Documentation</a>	39

5.11.2.1	iteration	39
5.11.2.2	sobol_jansen	39
5.11.2.3	sobol_martinez	39
5.11.2.4	variance_a	39
5.11.2.5	variance_b	39
5.12	sobol_jansen_s Struct Reference	39
5.12.1	Detailed Description	40
5.12.2	Member Data Documentation	40
5.12.2.1	first_order_values	40
5.12.2.2	summ_a	40
5.12.2.3	summ_b	40
5.12.2.4	total_order_values	40
5.13	sobol_martinez_s Struct Reference	41
5.13.1	Detailed Description	41
5.13.2	Member Data Documentation	41
5.13.2.1	confidence_interval	41
5.13.2.2	first_order_covariance	41
5.13.2.3	first_order_values	41
5.13.2.4	total_order_covariance	41
5.13.2.5	total_order_values	42
5.13.2.6	variance_k	42
5.14	variance_s Struct Reference	42
5.14.1	Detailed Description	42
5.14.2	Member Data Documentation	42
5.14.2.1	mean_structure	42
5.14.2.2	variance	43
5.15	zmq_data_s Struct Reference	43
5.15.1	Detailed Description	43
5.15.2	Member Data Documentation	43
5.15.2.1	buff_size	44



5.15.2.2	buffer	44
5.15.2.3	buffer_sobol	44
5.15.2.4	comm_sobol	44
5.15.2.5	connexion_requester	44
5.15.2.6	context	44
5.15.2.7	coupling	44
5.15.2.8	data_pusher	44
5.15.2.9	init_requester	45
5.15.2.10	local_nb_messages	45
5.15.2.11	local_vect_sizes	45
5.15.2.12	message_sizes	45
5.15.2.13	nb_parameters	45
5.15.2.14	nb_proc_server	45
5.15.2.15	pull_rank	45
5.15.2.16	push_rank	45
5.15.2.17	rinit_tab	46
5.15.2.18	sdispls	46
5.15.2.19	send_buff_size	46
5.15.2.20	send_counts	46
5.15.2.21	server_vect_size	46
5.15.2.22	sinit_tab	46
5.15.2.23	sobol	46
5.15.2.24	sobol_rank	46
5.15.2.25	sobol_requester	47
5.15.2.26	total_nb_messages	47

<b>6</b>	<b>File Documentation</b>	<b>49</b>
6.1	<a href="#">/home/tterraz/avido/melissa/Melissa/source/api/melissa_api.c File Reference</a>	49
6.2	<a href="#">/home/tterraz/avido/melissa/Melissa/source/api/melissa_api.h File Reference</a>	49
6.2.1	Detailed Description	49
6.3	<a href="#">/home/tterraz/avido/melissa/Melissa/source/api/melissa_api_no_mpi.h File Reference</a>	49
6.3.1	Detailed Description	50
6.4	<a href="#">/home/tterraz/avido/melissa/Melissa/source/server/compute_stats.c File Reference</a>	50
6.4.1	Detailed Description	50
6.5	<a href="#">/home/tterraz/avido/melissa/Melissa/source/server/compute_stats.h File Reference</a>	50
6.5.1	Detailed Description	51
6.6	<a href="#">/home/tterraz/avido/melissa/Melissa/source/server/fault_tolerance.c File Reference</a>	51
6.6.1	Detailed Description	51
6.7	<a href="#">/home/tterraz/avido/melissa/Melissa/source/server/fault_tolerance.h File Reference</a>	51
6.7.1	Detailed Description	52
6.8	<a href="#">/home/tterraz/avido/melissa/Melissa/source/server/melissa_data.c File Reference</a>	52
6.8.1	Detailed Description	53
6.8.2	Function Documentation	53
6.8.2.1	<a href="#">melissa_check_data()</a>	53
6.8.2.2	<a href="#">mem_conso()</a>	53
6.9	<a href="#">/home/tterraz/avido/melissa/Melissa/source/server/melissa_data.h File Reference</a>	53
6.9.1	Detailed Description	54
6.9.2	Typedef Documentation	54
6.9.2.1	<a href="#">comm_data_t</a>	54
6.9.2.2	<a href="#">melissa_data_t</a>	55
6.9.3	Function Documentation	55
6.9.3.1	<a href="#">melissa_check_data()</a>	55
6.9.3.2	<a href="#">mem_conso()</a>	55
6.10	<a href="#">/home/tterraz/avido/melissa/Melissa/source/server/melissa_fields.c File Reference</a>	55
6.10.1	Detailed Description	56
6.11	<a href="#">/home/tterraz/avido/melissa/Melissa/source/server/melissa_fields.h File Reference</a>	56

6.11.1 Detailed Description . . . . .	56
6.11.2 Typedef Documentation . . . . .	57
6.11.2.1 melissa_field_t . . . . .	57
6.12 /home/tterraz/avido/melissa/Melissa/source/server/melissa_io.c File Reference . . . . .	57
6.12.1 Detailed Description . . . . .	57
6.13 /home/tterraz/avido/melissa/Melissa/source/server/melissa_io.h File Reference . . . . .	58
6.13.1 Detailed Description . . . . .	58
6.14 /home/tterraz/avido/melissa/Melissa/source/server/melissa_options.c File Reference . . . . .	58
6.14.1 Detailed Description . . . . .	59
6.15 /home/tterraz/avido/melissa/Melissa/source/server/melissa_options.h File Reference . . . . .	59
6.15.1 Detailed Description . . . . .	60
6.15.2 Typedef Documentation . . . . .	60
6.15.2.1 melissa_options_t . . . . .	60
6.16 /home/tterraz/avido/melissa/Melissa/source/server/server.h File Reference . . . . .	60
6.16.1 Detailed Description . . . . .	61
6.16.2 Typedef Documentation . . . . .	61
6.16.2.1 pull_data_t . . . . .	61
6.16.3 Function Documentation . . . . .	61
6.16.3.1 global_confidence_sobol_martinez() . . . . .	61
6.17 /home/tterraz/avido/melissa/Melissa/source/stats/covariance.c File Reference . . . . .	62
6.17.1 Detailed Description . . . . .	62
6.17.2 Function Documentation . . . . .	62
6.17.2.1 free_covariance() . . . . .	62
6.17.2.2 increment_covariance() . . . . .	63
6.17.2.3 init_covariance() . . . . .	63
6.17.2.4 read_covariance() . . . . .	63
6.17.2.5 save_covariance() . . . . .	64
6.17.2.6 update_covariance() . . . . .	64
6.18 /home/tterraz/avido/melissa/Melissa/source/stats/covariance.h File Reference . . . . .	65
6.18.1 Detailed Description . . . . .	65

6.18.2	Typedef Documentation . . . . .	65
6.18.2.1	covariance_t . . . . .	65
6.18.3	Function Documentation . . . . .	65
6.18.3.1	free_covariance() . . . . .	65
6.18.3.2	increment_covariance() . . . . .	66
6.18.3.3	init_covariance() . . . . .	66
6.18.3.4	read_covariance() . . . . .	66
6.18.3.5	save_covariance() . . . . .	67
6.18.3.6	update_covariance() . . . . .	67
6.19	/home/tterraz/avido/melissa/Melissa/source/stats/mean.c File Reference . . . . .	68
6.19.1	Detailed Description . . . . .	68
6.19.2	Function Documentation . . . . .	68
6.19.2.1	free_mean() . . . . .	68
6.19.2.2	increment_mean() . . . . .	69
6.19.2.3	init_mean() . . . . .	69
6.19.2.4	read_mean() . . . . .	69
6.19.2.5	save_mean() . . . . .	70
6.19.2.6	update_mean() . . . . .	70
6.20	/home/tterraz/avido/melissa/Melissa/source/stats/mean.h File Reference . . . . .	70
6.20.1	Detailed Description . . . . .	71
6.20.2	Typedef Documentation . . . . .	71
6.20.2.1	mean_t . . . . .	71
6.20.3	Function Documentation . . . . .	71
6.20.3.1	free_mean() . . . . .	71
6.20.3.2	increment_mean() . . . . .	72
6.20.3.3	init_mean() . . . . .	72
6.20.3.4	read_mean() . . . . .	72
6.20.3.5	save_mean() . . . . .	73
6.20.3.6	update_mean() . . . . .	73
6.21	/home/tterraz/avido/melissa/Melissa/source/stats/min_max.c File Reference . . . . .	74

6.21.1 Detailed Description . . . . .	74
6.21.2 Function Documentation . . . . .	74
6.21.2.1 free_min_max() . . . . .	74
6.21.2.2 init_min_max() . . . . .	75
6.21.2.3 min_and_max() . . . . .	75
6.21.2.4 read_min_max() . . . . .	75
6.21.2.5 save_min_max() . . . . .	76
6.22 /home/tterraz/avido/melissa/Melissa/source/stats/min_max.h File Reference . . . . .	76
6.22.1 Detailed Description . . . . .	76
6.22.2 Typedef Documentation . . . . .	77
6.22.2.1 min_max_t . . . . .	77
6.22.3 Function Documentation . . . . .	77
6.22.3.1 free_min_max() . . . . .	77
6.22.3.2 init_min_max() . . . . .	77
6.22.3.3 min_and_max() . . . . .	77
6.22.3.4 read_min_max() . . . . .	78
6.22.3.5 save_min_max() . . . . .	78
6.23 /home/tterraz/avido/melissa/Melissa/source/stats/quantile.c File Reference . . . . .	79
6.23.1 Detailed Description . . . . .	79
6.23.2 Function Documentation . . . . .	79
6.23.2.1 free_quantile() . . . . .	79
6.23.2.2 increment_quantile() . . . . .	80
6.23.2.3 init_quantile() . . . . .	80
6.23.2.4 read_quantile() . . . . .	80
6.23.2.5 save_quantile() . . . . .	81
6.24 /home/tterraz/avido/melissa/Melissa/source/stats/quantile.h File Reference . . . . .	81
6.24.1 Detailed Description . . . . .	82
6.24.2 Typedef Documentation . . . . .	82
6.24.2.1 quantile_t . . . . .	82
6.24.3 Function Documentation . . . . .	82

6.24.3.1	<a href="#">free_quantile()</a> . . . . .	82
6.24.3.2	<a href="#">increment_quantile()</a> . . . . .	82
6.24.3.3	<a href="#">init_quantile()</a> . . . . .	83
6.24.3.4	<a href="#">read_quantile()</a> . . . . .	83
6.24.3.5	<a href="#">save_quantile()</a> . . . . .	83
6.25	<a href="#">/home/tterraz/avido/melissa/Melissa/source/stats/sobol.c File Reference</a> . . . . .	84
6.25.1	<a href="#">Detailed Description</a> . . . . .	85
6.25.2	<a href="#">Function Documentation</a> . . . . .	85
6.25.2.1	<a href="#">check_convergence_sobol_martinez()</a> . . . . .	85
6.25.2.2	<a href="#">confidence_sobol_martinez()</a> . . . . .	85
6.25.2.3	<a href="#">free_sobol_jansen()</a> . . . . .	86
6.25.2.4	<a href="#">free_sobol_martinez()</a> . . . . .	86
6.25.2.5	<a href="#">increment_sobol_jansen()</a> . . . . .	86
6.25.2.6	<a href="#">increment_sobol_martinez()</a> . . . . .	87
6.25.2.7	<a href="#">init_sobol_jansen()</a> . . . . .	87
6.25.2.8	<a href="#">init_sobol_martinez()</a> . . . . .	87
6.25.2.9	<a href="#">read_sobol_jansen()</a> . . . . .	88
6.25.2.10	<a href="#">read_sobol_martinez()</a> . . . . .	88
6.25.2.11	<a href="#">save_sobol_jansen()</a> . . . . .	89
6.25.2.12	<a href="#">save_sobol_martinez()</a> . . . . .	89
6.26	<a href="#">/home/tterraz/avido/melissa/Melissa/source/stats/sobol.h File Reference</a> . . . . .	90
6.26.1	<a href="#">Detailed Description</a> . . . . .	90
6.26.2	<a href="#">Typedef Documentation</a> . . . . .	91
6.26.2.1	<a href="#">sobol_array_t</a> . . . . .	91
6.26.2.2	<a href="#">sobol_jansen_t</a> . . . . .	91
6.26.2.3	<a href="#">sobol_martinez_t</a> . . . . .	91
6.26.3	<a href="#">Function Documentation</a> . . . . .	91
6.26.3.1	<a href="#">check_convergence_sobol_martinez()</a> . . . . .	91
6.26.3.2	<a href="#">confidence_sobol_martinez()</a> . . . . .	92
6.26.3.3	<a href="#">free_sobol_jansen()</a> . . . . .	92

6.26.3.4	<a href="#">free_sobol_martinez()</a>	92
6.26.3.5	<a href="#">increment_sobol_jansen()</a>	93
6.26.3.6	<a href="#">increment_sobol_martinez()</a>	93
6.26.3.7	<a href="#">init_sobol_jansen()</a>	93
6.26.3.8	<a href="#">init_sobol_martinez()</a>	94
6.26.3.9	<a href="#">read_sobol_jansen()</a>	94
6.26.3.10	<a href="#">read_sobol_martinez()</a>	95
6.26.3.11	<a href="#">save_sobol_jansen()</a>	95
6.26.3.12	<a href="#">save_sobol_martinez()</a>	95
6.27	<a href="#">/home/tterraz/avido/melissa/Melissa/source/stats/threshold.c File Reference</a>	97
6.27.1	Detailed Description	97
6.27.2	Function Documentation	97
6.27.2.1	<a href="#">read_threshold()</a>	97
6.27.2.2	<a href="#">save_threshold()</a>	98
6.27.2.3	<a href="#">update_threshold_exceedance()</a>	98
6.28	<a href="#">/home/tterraz/avido/melissa/Melissa/source/stats/threshold.h File Reference</a>	99
6.28.1	Detailed Description	99
6.28.2	Function Documentation	99
6.28.2.1	<a href="#">read_threshold()</a>	99
6.28.2.2	<a href="#">save_threshold()</a>	99
6.28.2.3	<a href="#">update_threshold_exceedance()</a>	100
6.29	<a href="#">/home/tterraz/avido/melissa/Melissa/source/stats/variance.c File Reference</a>	100
6.29.1	Detailed Description	101
6.29.2	Function Documentation	101
6.29.2.1	<a href="#">free_variance()</a>	101
6.29.2.2	<a href="#">increment_mean_and_variance()</a>	101
6.29.2.3	<a href="#">increment_variance()</a>	102
6.29.2.4	<a href="#">init_variance()</a>	102
6.29.2.5	<a href="#">read_variance()</a>	102
6.29.2.6	<a href="#">save_variance()</a>	103

6.29.2.7	<a href="#">update_variance()</a>	103
6.30	<a href="#">/home/tterraz/avido/melissa/Melissa/source/stats/variance.h File Reference</a>	104
6.30.1	Detailed Description	104
6.30.2	Typedef Documentation	104
6.30.2.1	<a href="#">variance_t</a>	104
6.30.3	Function Documentation	104
6.30.3.1	<a href="#">free_variance()</a>	104
6.30.3.2	<a href="#">increment_mean_and_variance()</a>	105
6.30.3.3	<a href="#">increment_variance()</a>	105
6.30.3.4	<a href="#">init_variance()</a>	105
6.30.3.5	<a href="#">read_variance()</a>	106
6.30.3.6	<a href="#">save_variance()</a>	106
6.30.3.7	<a href="#">update_variance()</a>	106
6.31	<a href="#">/home/tterraz/avido/melissa/Melissa/source/utils/melissa_utils.c File Reference</a>	107
6.31.1	Detailed Description	108
6.32	<a href="#">/home/tterraz/avido/melissa/Melissa/source/utils/melissa_utils.h File Reference</a>	108
6.32.1	Detailed Description	108
6.32.2	Macro Definition Documentation	109
6.32.2.1	<a href="#">MAX_FIELD_NAME</a>	109
6.32.2.2	<a href="#">MPI_MAX_PROCESSOR_NAME</a>	109
<b>Index</b>		<b>111</b>



# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

misc functions . . . . .	7
internal API . . . . .	11
Melissa fields . . . . .	12
Melissa data . . . . .	13
input, output and checkpoint functions . . . . .	14
Get options from command line . . . . .	19
API . . . . .	21



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">comm_data_s</a>	25
<a href="#">covariance_s</a>	26
<a href="#">mean_s</a>	27
<a href="#">melissa_data_s</a>	28
<a href="#">melissa_field_s</a>	31
<a href="#">melissa_options_s</a>	32
<a href="#">melissa_simulation_s</a>	35
<a href="#">min_max_s</a>	35
<a href="#">pull_data_s</a>	36
<a href="#">quantile_s</a>	37
<a href="#">sobol_array_s</a>	38
<a href="#">sobol_jansen_s</a>	39
<a href="#">sobol_martinez_s</a>	41
<a href="#">variance_s</a>	42
<a href="#">zmq_data_s</a>	43



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

/home/tterraz/avido/melissa/Melissa/source/api/ <a href="#">melissa_api.c</a>	
API Functions . . . . .	49
/home/tterraz/avido/melissa/Melissa/source/api/ <a href="#">melissa_api.h</a> . . . . .	49
/home/tterraz/avido/melissa/Melissa/source/api/ <a href="#">melissa_api_no_mpi.h</a> . . . . .	49
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">compute_stats.c</a>	
Functions called by the server . . . . .	50
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">compute_stats.h</a> . . . . .	50
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">fault_tolerance.c</a> . . . . .	51
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">fault_tolerance.h</a> . . . . .	51
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">melissa_data.c</a>	
Routines related to the melissa_data structure . . . . .	52
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">melissa_data.h</a> . . . . .	53
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">melissa_fields.c</a>	
Routines related to the melissa_fields structure . . . . .	55
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">melissa_fields.h</a> . . . . .	56
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">melissa_io.c</a>	
Inputs, outputs and checkpoints . . . . .	57
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">melissa_io.h</a> . . . . .	58
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">melissa_options.c</a>	
Parse commande line to get stats options . . . . .	58
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">melissa_options.h</a> . . . . .	59
/home/tterraz/avido/melissa/Melissa/source/server/ <a href="#">server.h</a> . . . . .	60
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">covariance.c</a>	
Functions needed to compute covariances . . . . .	62
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">covariance.h</a> . . . . .	65
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">mean.c</a>	
Mean related functions . . . . .	68
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">mean.h</a> . . . . .	70
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">min_max.c</a>	
Min and max related functions . . . . .	74
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">min_max.h</a> . . . . .	76
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">quantile.c</a>	
Quantile related functions . . . . .	79
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">quantile.h</a> . . . . .	81
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">sobol.c</a>	
Functions needed to compute sobol indices . . . . .	84

/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">sobol.h</a> . . . . .	90
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">threshold.c</a>	
Threshold exceedance related functions . . . . .	97
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">threshold.h</a> . . . . .	99
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">variance.c</a>	
Variance related functions . . . . .	100
/home/tterraz/avido/melissa/Melissa/source/stats/ <a href="#">variance.h</a> . . . . .	104
/home/tterraz/avido/melissa/Melissa/source/utils/ <a href="#">melissa_utils.c</a>	
Functions used in Melissa . . . . .	107
/home/tterraz/avido/melissa/Melissa/source/utils/ <a href="#">melissa_utils.h</a> . . . . .	108

# Chapter 4

## Module Documentation

### 4.1 misc functions

#### Functions

- void [melissa\\_logo](#) ()
- void \* [melissa\\_malloc](#) (size\_t size)
- void \* [melissa\\_calloc](#) (size\_t num, size\_t size)
- void [melissa\\_free](#) (void \*ptr)
- void [melissa\\_bind](#) (void \*socket, char \*port\_name)
- void [melissa\\_connect](#) (void \*socket, char \*port\_name)
- double [melissa\\_get\\_time](#) ()
- void [melissa\\_get\\_node\\_name](#) (char \*node\_name)
- void [set\\_bit](#) (int32\_t \*vect, int pos)
- void [clear\\_bit](#) (int32\_t \*vect, int pos)
- int [test\\_bit](#) (int32\_t \*vect, int pos)

#### 4.1.1 Detailed Description

#### 4.1.2 Function Documentation

##### 4.1.2.1 clear\_bit()

```
void clear_bit (  
    int32_t * vect,  
    int pos )
```

Sets a bit to 0 in an array of bits

#### Parameters

out	<i>*vect</i>	pointer to the array of bits
in	<i>*pos</i>	position in the array of bits to set to 0

#### 4.1.2.2 melissa\_bind()

```
void melissa_bind (
    void * socket,
    char * port_name )
```

Wrapper around zmq\_bind

##### Parameters

in	<i>*socket</i>	ZMQ socket handler
in	<i>port_name</i>	Port name

#### 4.1.2.3 melissa\_calloc()

```
void* melissa_calloc (
    size_t num,
    size_t size )
```

Wrapper around melissa\_calloc

##### Parameters

in	<i>num</i>	Number of elements to allocate
in	<i>size</i>	Size of one element

##### Returns

The pointer to the allocated memory

#### 4.1.2.4 melissa\_connect()

```
void melissa_connect (
    void * socket,
    char * port_name )
```

Wrapper around zmq\_connect

##### Parameters

in	<i>*socket</i>	ZMQ socket handler
in	<i>port_name</i>	Port name



#### 4.1.2.5 melissa\_free()

```
void melissa_free (
    void * ptr )
```

Free and nullify a pointer

##### Parameters

in	*ptr	The pointer to free and nullify
----	------	---------------------------------

#### 4.1.2.6 melissa\_get\_node\_name()

```
void melissa_get_node_name (
    char * node_name )
```

Gets the name of the processus node

##### Parameters

out	*node_name	The node name
-----	------------	---------------

#### 4.1.2.7 melissa\_get\_time()

```
double melissa_get_time ( )
```

Returns an elapsed time

##### Returns

elapsed time

#### 4.1.2.8 melissa\_logo()

```
void melissa_logo ( )
```

Prints Melissa logo

#### 4.1.2.9 melissa\_malloc()

```
void* melissa_malloc (
    size_t size )
```

Wrapper around melissa\_malloc

**Parameters**

in	<i>size</i>	Number of bytes to allocate
----	-------------	-----------------------------

**Returns**

The pointer to the allocated memory

**4.1.2.10 set\_bit()**

```
void set_bit (
    int32_t * vect,
    int pos )
```

Sets a bit to 1 in an array of bits

**Parameters**

out	<i>*vect</i>	pointer to the array of bits
in	<i>*pos</i>	position in the array of bits to set to 1

**4.1.2.11 test\_bit()**

```
int test_bit (
    int32_t * vect,
    int pos )
```

Tests a value in an array of bits

**Parameters**

out	<i>*vect</i>	pointer to the array of bits
in	<i>*pos</i>	position to test in the array of bits

## 4.2 internal API

### Functions

- void `compute_stats` (`melissa_data_t` \*data, const int time\_step, const int nb\_vect, double \*\*in\_vect\_tab, const int group\_id)
- void `finalize_stats` (`melissa_data_t` \*data)

#### 4.2.1 Detailed Description

#### 4.2.2 Function Documentation

##### 4.2.2.1 `compute_stats()`

```
void compute_stats (
    melissa_data_t * data,
    const int time_step,
    const int nb_vect,
    double ** in_vect_tab,
    const int group_id )
```

This function updates the statistics stored in the data structure

##### Parameters

in	<i>*data</i>	pointer to the structure containing global parameters
in	<i>time_step</i>	time step of the current simulation
in	<i>nb_vect</i>	number of input vectors
in	<i>**in_vect_tab</i>	array of input vectors
in	<i>group_id</i>	ID of the simulation providing in_vect_tab

##### 4.2.2.2 `finalize_stats()`

```
void finalize_stats (
    melissa_data_t * data )
```

This function finalize the statistics stored in the data structure

##### Parameters

in	<i>*data</i>	pointer to the structure containing global parameters
----	--------------	---

### 4.3 Melissa fields

## 4.4 Melissa data

### Functions

- void `melissa_init_data` (`melissa_data_t` \*data, `melissa_options_t` \*options, int vect\_size)
- void `melissa_free_data` (`melissa_data_t` \*data)

#### 4.4.1 Detailed Description

#### 4.4.2 Function Documentation

##### 4.4.2.1 `melissa_free_data()`

```
void melissa_free_data (  
    melissa_data_t * data )
```

This function frees the memory in the data structure

##### Parameters

in	*data	pointer to the structure containing global parameters
----	-------	---

##### 4.4.2.2 `melissa_init_data()`

```
void melissa_init_data (  
    melissa_data_t * data,  
    melissa_options_t * options,  
    int vect_size )
```

This function initializes the data structure

##### Parameters

out	*data	pointer to the structure containing global parameters
in	*options	pointer to the structure containing options parsed from command line
in	vect_size	size of the local input vector

## 4.5 input, output and checkpoint functions

### Functions

- void [write\\_client\\_data](#) (int \*client\_comm\_size, int \*client\_vect\_sizes)
- int [read\\_client\\_data](#) (int \*client\_comm\_size, int \*\*client\_vect\_sizes, [melissa\\_options\\_t](#) \*options)
- void [save\\_stats](#) ([melissa\\_data\\_t](#) \*data, [comm\\_data\\_t](#) \*comm\_data, char \*field\_name)
- void [read\\_saved\\_stats](#) ([melissa\\_data\\_t](#) \*data, [comm\\_data\\_t](#) \*comm\_data, char \*field\_name, int client\_rank)
- void [save\\_simu\\_states](#) (int \*simu\_states, [comm\\_data\\_t](#) \*comm\_data, int size)
- void [read\\_simu\\_states](#) (int \*simu\_states, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int size)
- void [write\\_stats\\_bin](#) ([melissa\\_data\\_t](#) \*\*data, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int \*local\_vect\_sizes, char \*field)
- void [write\\_stats\\_ensight](#) ([melissa\\_data\\_t](#) \*\*data, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int \*local\_vect\_sizes, char \*field)
- void [read\\_ensight](#) ([melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, double \*in\_vect, int \*local\_vect\_sizes, char \*file\_name)
- void [write\\_stats\\_txt](#) ([melissa\\_data\\_t](#) \*\*data, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int \*local\_vect\_sizes, char \*field)

### 4.5.1 Detailed Description

### 4.5.2 Function Documentation

#### 4.5.2.1 read\_client\_data()

```
int read_client_data (
    int * client_comm_size,
    int ** client_vect_sizes,
    melissa\_options\_t * options )
```

This function reads a saved option structure on disc

#### Parameters

out	* <i>client_comm_size</i>	client MPI communicator size
out	** <i>client_vect_sizes</i>	client vector sizes
in	* <i>options</i>	Melissa option structure

#### 4.5.2.2 read\_ensight()

```
void read_ensight (
    melissa\_options\_t * options,
    comm\_data\_t * comm_data,
    double * in_vect,
```

```
int * local_vect_sizes,
char * file_name )
```

This function reads data from Ensign files

#### Parameters

in	<i>*options</i>	option structure
in	<i>*comm_data</i>	structure containing communications parameters
in	<i>*in_vect</i>	input vector
in	<i>*local_vect_sizes</i>	all local vector sizes
in	<i>*file_name</i>	name of the file

#### 4.5.2.3 read\_saved\_stats()

```
void read_saved_stats (
    melissa_data_t * data,
    comm_data_t * comm_data,
    char * field_name,
    int client_rank )
```

This function reads stats saved on disc

#### Parameters

in	<i>*data</i>	data structure to read
in	<i>*comm_data</i>	communication structure
in	<i>*field_name</i>	name of the field to read
in	<i>client_rank</i>	mpi rank of sending client process

#### 4.5.2.4 read\_simu\_states()

```
void read_simu_states (
    int * simu_states,
    melissa_options_t * options,
    comm_data_t * comm_data,
    int size )
```

This function reads simulation states from disc

#### Parameters

out	<i>*simu_states</i>	array of simulation states
in	<i>*options</i>	Melissa option structure
in	<i>*comm_data</i>	communication structure
in	<i>size</i>	size of *simu_states

#### 4.5.2.5 save\_simu\_states()

```
void save_simu_states (
    int * simu_states,
    comm_data_t * comm_data,
    int size )
```

This function saves current simulation states on disc

##### Parameters

in	<i>*simu_states</i>	array of simulation states
in	<i>*comm_data</i>	communication structure
in	<i>size</i>	size of <i>*simu_states</i>

#### 4.5.2.6 save\_stats()

```
void save_stats (
    melissa_data_t * data,
    comm_data_t * comm_data,
    char * field_name )
```

This function saves stats on disc

##### Parameters

in	<i>*data</i>	data structure to save
in	<i>*comm_data</i>	communication structure
in	<i>*field_name</i>	name of the field to write

#### 4.5.2.7 write\_client\_data()

```
void write_client_data (
    int * client_comm_size,
    int * client_vect_sizes )
```

This function saves some client data on disc

##### Parameters

in	<i>*client_comm_size</i>	client MPI communicator size
in	<i>*client_vect_sizes</i>	client vector sizes



#### 4.5.2.8 write\_stats\_bin()

```
void write_stats_bin (
    melissa_data_t ** data,
    melissa_options_t * options,
    comm_data_t * comm_data,
    int * local_vect_sizes,
    char * field )
```

This function writes the computed statistics on files

##### Parameters

in	** <i>data</i>	pointer to the array of structures containing statistics data
in	* <i>options</i>	Melissa option structure
in	<i>comm_data</i>	structure containing communications parameters
in	* <i>local_vect_sizes</i>	all local vector sizes
in	* <i>field</i>	name of the field on which are computed the statistics

#### 4.5.2.9 write\_stats\_ensight()

```
void write_stats_ensight (
    melissa_data_t ** data,
    melissa_options_t * options,
    comm_data_t * comm_data,
    int * local_vect_sizes,
    char * field )
```

This function writes the computed statistics on files

##### Parameters

in	** <i>data</i>	pointer to the array of structures containing statistics data
in	* <i>options</i>	option structure
in	* <i>comm_data</i>	structure containing communications parameters
in	* <i>local_vect_sizes</i>	all local vector sizes
in	* <i>field</i>	name of the field on which are computed the statistics

#### 4.5.2.10 write\_stats\_txt()

```
void write_stats_txt (
    melissa_data_t ** data,
```

```
melissa_options_t * options,  
comm_data_t * comm_data,  
int * local_vect_sizes,  
char * field )
```

This function writes the computed statistics on files

#### Parameters

in	<b><i>**data</i></b>	pointer to the array of structures containing statistics data
in	<b><i>*options</i></b>	Melissa option structure
in	<b><i>comm_data</i></b>	structure containing communications parameters
in	<b><i>*local_vect_sizes</i></b>	all local vector sizes
in	<b><i>*field</i></b>	name of the field on which are computed the statistics

## 4.6 Get options from command line

### Functions

- void [melissa\\_print\\_options](#) ([melissa\\_options\\_t](#) \*options)
- void [melissa\\_get\\_options](#) (int argc, char \*\*argv, [melissa\\_options\\_t](#) \*options)
- void [melissa\\_check\\_options](#) ([melissa\\_options\\_t](#) \*options)
- void [melissa\\_write\\_options](#) ([melissa\\_options\\_t](#) \*options)
- int [melissa\\_read\\_options](#) ([melissa\\_options\\_t](#) \*options)

### 4.6.1 Detailed Description

### 4.6.2 Function Documentation

#### 4.6.2.1 [melissa\\_check\\_options\(\)](#)

```
void melissa_check_options (  
    melissa\_options\_t * options )
```

This function validates the option structure

#### Parameters

out	<i>*options</i>	pointer to the structure containing options parameters
-----	-----------------	--

#### 4.6.2.2 [melissa\\_get\\_options\(\)](#)

```
void melissa_get_options (  
    int argc,  
    char ** argv,  
    melissa\_options\_t * options )
```

This function parses command line options and fill the parameter structure

#### Parameters

in	<i>argc</i>	argc
in	<i>**argv</i>	argv
out	<i>*options</i>	pointer to the structure containing options parameters

#### 4.6.2.3 melissa\_print\_options()

```
void melissa_print_options (
    melissa_options_t * options )
```

This function displays the global parameters on stdout

##### Parameters

in	* <i>options</i>	pointer to the structure containing the options parsed from command line
----	------------------	--

#### 4.6.2.4 melissa\_read\_options()

```
int melissa_read_options (
    melissa_options_t * options )
```

This function reads a saved option structure on disc

##### Parameters

in, out	* <i>options</i>	pointer to the structure containing global options
---------	------------------	--

#### 4.6.2.5 melissa\_write\_options()

```
void melissa_write_options (
    melissa_options_t * options )
```

This function writes the option structure on disc

##### Parameters

in	* <i>options</i>	pointer to the structure containing global options
----	------------------	--

## 4.7 API

### Classes

- struct [zmq\\_data\\_s](#)

### Functions

- void [melissa\\_init](#) (const int \*local\_vect\_size, const int \*comm\_size, const int \*rank, const int \*sobol\_rank, const int \*sample\_id, [MPI\\_Comm](#) \*comm, const int \*coupling)
- void [melissa\\_init\\_no\\_mpi](#) (const int \*vect\_size, const int \*sobol\_rank, const int \*sample\_id)
- void [melissa\\_send](#) (const int \*time\_step, const char \*field\_name, double \*send\_vect, const int \*rank, const int \*sobol\_rank, const int \*sample\_id)
- void [melissa\\_send\\_no\\_mpi](#) (const int \*time\_step, const char \*field\_name, double \*send\_vect, const int \*sobol\_rank, const int \*sample\_id)
- void [melissa\\_finalize](#) ()

#### 4.7.1 Detailed Description

#### 4.7.2 Function Documentation

##### 4.7.2.1 [melissa\\_finalize\(\)](#)

```
void melissa_finalize ( )
```

This function disconnects the simulation from the statistic library

##### 4.7.2.2 [melissa\\_init\(\)](#)

```
void melissa_init (
    const int * local_vect_size,
    const int * comm_size,
    const int * rank,
    const int * sobol_rank,
    const int * sample_id,
    MPI\_Comm * comm,
    const int * coupling )
```

This function initialise connexion with the stats library

#### Parameters

in	<i>*local_vect_size</i>	size of the local data vector to send to the library
in	<i>*comm_size</i>	size of the MPI communicator comm
in	<i>*rank</i>	MPI rank
in	<i>*sobol_rank</i>	Sobol indice rank in Sobol group
in	<i>*sample_id</i>	ID of the parameter set defining the simulation
in	<i>*comm</i>	MPI communicator
in	<i>*coupling</i>	1 if simulation are coupled in the same MPI_COMM_WORLD, 0 otherwise

#### 4.7.2.3 melissa\_init\_no\_mpi()

```
void melissa_init_no_mpi (
    const int * vect_size,
    const int * sobol_rank,
    const int * sample_id )
```

This function initialise connexion with Melissa Server for sequential simulations

##### Parameters

in	<i>*vect_size</i>	size of the data vector to send to the library
in	<i>*sobol_rank</i>	Sobol indice rank in Sobol group
in	<i>*sample_id</i>	ID of the parameter set defining the simulation

#### 4.7.2.4 melissa\_send()

```
void melissa_send (
    const int * time_step,
    const char * field_name,
    double * send_vect,
    const int * rank,
    const int * sobol_rank,
    const int * sample_id )
```

This function sends data to Melissa Server

##### Parameters

in	<i>time_step</i>	current time step of the simulation
in	<i>*field_name</i>	name of the field to send to Melissa Server
in	<i>*send_vect</i>	local data array to send to the statistic library
in	<i>*rank</i>	MPI rank
in	<i>*sobol_rank</i>	Sobol indice rank in Sobol group
in	<i>*sample_id</i>	ID of the parameter set defining the simulation

#### 4.7.2.5 melissa\_send\_no\_mpi()

```
void melissa_send_no_mpi (
    const int * time_step,
    const char * field_name,
    double * send_vect,
```

```
const int * sobol_rank,  
const int * sample_id )
```

This function sends data to the stats library

#### Parameters

in	<i>time_step</i>	current time step of the simulation
in	<i>*field_name</i>	name of the field to send to Melissa Server
in	<i>*send_vect</i>	local data array to send to the statistic library
in	<i>*sobol_rank</i>	Sobol indice rank in Sobol group
in	<i>*sample_id</i>	ID of the parameter set defining the simulation





## Chapter 5

# Class Documentation

### 5.1 comm\_data\_s Struct Reference

```
#include <melissa_data.h>
```

#### Public Attributes

- int [rank](#)
- int [comm\\_size](#)
- int [client\\_comm\\_size](#)
- int \* [rcounts](#)
- int \* [rdispls](#)

#### 5.1.1 Detailed Description

Structure to store communications parameters

#### 5.1.2 Member Data Documentation

##### 5.1.2.1 client\_comm\_size

```
int comm_data_s::client_comm_size
```

size of the clients communicators

##### 5.1.2.2 comm\_size

```
int comm_data_s::comm_size
```

size of the MPI communicator (1 if sequential)

### 5.1.2.3 rank

```
int comm_data_s::rank
```

rank of the MPI process (0 if sequential)

### 5.1.2.4 rcounts

```
int* comm_data_s::rcounts
```

counts for receiving datas

### 5.1.2.5 rdispls

```
int* comm_data_s::rdispls
```

displacements for receiving datas

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

- [/home/tterraz/avido/melissa/Melissa/source/server/melissa\\_data.h](#)

## 5.2 covariance\_s Struct Reference

```
#include <covariance.h>
```

Collaboration diagram for covariance\_s:

### Public Attributes

- double \* [covariance](#)
- [mean\\_t](#) mean1
- [mean\\_t](#) mean2
- int [increment](#)

### 5.2.1 Detailed Description

Structure containing an array of covariances and the corresponding mean structures

### 5.2.2 Member Data Documentation

#### 5.2.2.1 covariance

```
double* covariance_s::covariance
```

```
covariance[vect_size]
```

#### 5.2.2.2 increment

```
int covariance_s::increment
```

```
increment
```

#### 5.2.2.3 mean1

```
mean_t covariance_s::mean1
```

corresponding mean

#### 5.2.2.4 mean2

```
mean_t covariance_s::mean2
```

corresponding mean

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

- [/home/tterraz/avido/melissa/Melissa/source/stats/covariance.h](#)

## 5.3 mean\_s Struct Reference

```
#include <mean.h>
```

### Public Attributes

- double \* [mean](#)
- int [increment](#)

#### 5.3.1 Detailed Description

Structure containing an array of means, and the corresponding increment

#### 5.3.2 Member Data Documentation

### 5.3.2.1 increment

```
int mean_s::increment
```

increment of this mean

### 5.3.2.2 mean

```
double* mean_s::mean
```

```
mean[vect_size]
```

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

- [/home/tterraz/avido/melissa/Melissa/source/stats/mean.h](#)

## 5.4 melissa\_data\_s Struct Reference

```
#include <melissa_data.h>
```

Collaboration diagram for melissa\_data\_s:

### Public Attributes

- int [vect\\_size](#)
- [melissa\\_options\\_t](#) \* [options](#)
- int [is\\_valid](#)
- [mean\\_t](#) \* [means](#)
- [variance\\_t](#) \* [variances](#)
- [min\\_max\\_t](#) \* [min\\_max](#)
- int \*\* [thresholds](#)
- [quantile\\_t](#) \* [quantiles](#)
- [sobol\\_array\\_t](#) \* [sobol\\_indices](#)
- void(\* [init\\_sobol](#) )([sobol\\_array\\_t](#) \*, int, int)
- void(\* [read\\_sobol](#) )([sobol\\_array\\_t](#) \*, int, int, int, FILE \*)
- void(\* [save\\_sobol](#) )([sobol\\_array\\_t](#) \*, int, int, int, FILE \*)
- void(\* [increment\\_sobol](#) )([sobol\\_array\\_t](#) \*, int, double \*\*, int)
- void(\* [free\\_sobol](#) )([sobol\\_array\\_t](#) \*, int)
- int [nb\\_simu](#)
- int32\_t \*\* [step\\_simu](#)

### 5.4.1 Detailed Description

Structure to store global parameters

### 5.4.2 Member Data Documentation

#### 5.4.2.1 free\_sobol

```
void(* melissa_data_s::free_sobol) (sobol_array_t *, int)
```

pointer to Sobol free function

#### 5.4.2.2 increment\_sobol

```
void(* melissa_data_s::increment_sobol) (sobol_array_t *, int, double **, int)
```

pointer to Sobol increment function

#### 5.4.2.3 init\_sobol

```
void(* melissa_data_s::init_sobol) (sobol_array_t *, int, int)
```

pointer to Sobol initialization function

#### 5.4.2.4 is\_valid

```
int melissa_data_s::is_valid
```

1 if the structure has been checked

#### 5.4.2.5 means

```
mean_t* melissa_data_s::means
```

array of mean structures, size nb\_time\_steps

#### 5.4.2.6 min\_max

```
min_max_t* melissa_data_s::min_max
```

array of min and max structures, size nb\_time\_steps

#### 5.4.2.7 nb\_simu

```
int melissa_data_s::nb_simu
```

number of simulation that have sent a message

#### 5.4.2.8 options

```
melissa_options_t* melissa_data_s::options
```

pointer to an option structure

#### 5.4.2.9 quantiles

```
quantile_t* melissa_data_s::quantiles
```

array of quantile structures, size nb\_time\_steps

#### 5.4.2.10 read\_sobol

```
void(* melissa_data_s::read_sobol) (sobol_array_t *, int, int, int, FILE *)
```

pointer to Sobol read function

#### 5.4.2.11 save\_sobol

```
void(* melissa_data_s::save_sobol) (sobol_array_t *, int, int, int, FILE *)
```

pointer to Sobol save function

#### 5.4.2.12 sobol\_indices

```
sobol_array_t* melissa_data_s::sobol_indices
```

array of sobol array structures, size nb\_time\_steps

#### 5.4.2.13 step\_simu

```
int32_t** melissa_data_s::step_simu
```

arrays of bits, size nb\_groups

#### 5.4.2.14 thresholds

```
int** melissa_data_s::thresholds
```

array of threshold exceedance vectors, size nb\_time\_steps

#### 5.4.2.15 variances

```
variance_t* melissa_data_s::variances
```

array of variance structures, size nb\_time\_steps

#### 5.4.2.16 vect\_size

```
int melissa_data_s::vect_size
```

local size of input vectors

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

- [/home/tterraz/avido/melissa/Melissa/source/server/melissa\\_data.h](#)

## 5.5 melissa\_field\_s Struct Reference

```
#include <melissa_fields.h>
```

Collaboration diagram for melissa\_field\_s:

### Public Attributes

- char [name](#) [[MAX\\_FIELD\\_NAME](#)]
- [melissa\\_data\\_t](#) \* [stats\\_data](#)

### 5.5.1 Detailed Description

< Structure for a linked list of output fields

Field structure

### 5.5.2 Member Data Documentation

#### 5.5.2.1 name

```
char melissa_field_s::name[MAX\_FIELD\_NAME]
```

name of the field

#### 5.5.2.2 stats\_data

```
melissa\_data\_t* melissa_field_s::stats_data
```

stats\_data structure

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

- [/home/tterraz/avido/melissa/Melissa/source/server/melissa\\_fields.h](#)

## 5.6 melissa\_options\_s Struct Reference

```
#include <melissa_options.h>
```

### Public Attributes

- int [nb\\_time\\_steps](#)
- int [nb\\_parameters](#)
- int [sampling\\_size](#)
- int [nb\\_simu](#)
- int [nb\\_fields](#)
- int [mean\\_op](#)
- int [variance\\_op](#)
- int [min\\_and\\_max\\_op](#)
- int [threshold\\_op](#)
- double [threshold](#)
- int [quantile\\_op](#)
- int [sobol\\_op](#)
- int [sobol\\_order](#)
- int [global\\_vect\\_size](#)
- int [restart](#)
- char [restart\\_dir](#) [256]
- char [launcher\\_name](#) [256]

### 5.6.1 Detailed Description

Structure to store options parsed from command line

### 5.6.2 Member Data Documentation

#### 5.6.2.1 global\_vect\_size

```
int melissa_options_s::global_vect_size
```

global size of input vector

#### 5.6.2.2 launcher\_name

```
char melissa_options_s::launcher_name[256]
```

Melissa master node name



### 5.6.2.3 mean\_op

```
int melissa_options_s::mean_op
```

1 if the user needs to compute the means, 0 otherwise.

### 5.6.2.4 min\_and\_max\_op

```
int melissa_options_s::min_and_max_op
```

1 if the user needs to compute min and max, 0 otherwise.

### 5.6.2.5 nb\_fields

```
int melissa_options_s::nb_fields
```

nb of fields of the simulations

### 5.6.2.6 nb\_parameters

```
int melissa_options_s::nb_parameters
```

nb of variables parameters of the study

### 5.6.2.7 nb\_simu

```
int melissa_options_s::nb_simu
```

nb of simulation of the study

### 5.6.2.8 nb\_time\_steps

```
int melissa_options_s::nb_time_steps
```

number of time steps of the study

### 5.6.2.9 quantile\_op

```
int melissa_options_s::quantile_op
```

1 if the user needs to compute quantiles, 0 otherwise

### 5.6.2.10 restart

```
int melissa_options_s::restart
```

1 if restart, 0 otherwise

#### 5.6.2.11 restart\_dir

```
char melissa_options_s::restart_dir[256]
```

Melissa restart files directory

#### 5.6.2.12 sampling\_size

```
int melissa_options_s::sampling_size
```

nb of randomly drawn simulation parameter sets

#### 5.6.2.13 sobol\_op

```
int melissa_options_s::sobol_op
```

1 if the user needs to compute sobol indices, 0 otherwise

#### 5.6.2.14 sobol\_order

```
int melissa_options_s::sobol_order
```

max order of the computes sobol indices

#### 5.6.2.15 threshold

```
double melissa_options_s::threshold
```

threshold used to compute threshold exceedance

#### 5.6.2.16 threshold\_op

```
int melissa_options_s::threshold_op
```

1 if the user needs to compute threshold exceedance, 0 otherwise

#### 5.6.2.17 variance\_op

```
int melissa_options_s::variance_op
```

1 if the user needs to compute the variances, 0 otherwise.

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

- [/home/tterraz/avido/melissa/Melissa/source/server/melissa\\_options.h](#)

## 5.7 melissa\_simulation\_s Struct Reference

### Public Attributes

- int **id**
- int **status**
- int **timeout**
- int **last\_message**

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

- /home/tterraz/avido/melissa/Melissa/source/server/[fault\\_tolerance.h](#)

## 5.8 min\_max\_s Struct Reference

```
#include <min_max.h>
```

### Public Attributes

- double \* [min](#)
- double \* [max](#)
- int [is\\_init](#)

### 5.8.1 Detailed Description

Structure containing two arrays of min and max values

### 5.8.2 Member Data Documentation

#### 5.8.2.1 is\_init

```
int min_max_s::is_init
```

0 before the first update, 1 otherwise

#### 5.8.2.2 max

```
double* min_max_s::max
```

```
max[vect_size]
```

### 5.8.2.3 min

```
double* min_max_s::min
```

```
min[vect_size]
```

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

- [/home/tterraz/avido/melissa/Melissa/source/stats/min\\_max.h](/home/tterraz/avido/melissa/Melissa/source/stats/min_max.h)

## 5.9 pull\_data\_s Struct Reference

```
#include <server.h>
```

### Public Attributes

- int \* [pull\\_rank](#)
- int \* [push\\_rank](#)
- int \* [message\\_sizes](#)
- int [total\\_nb\\_messages](#)
- int [local\\_nb\\_messages](#)
- int [buff\\_size](#)

### 5.9.1 Detailed Description

< Helper structure for push pull socket

### 5.9.2 Member Data Documentation

#### 5.9.2.1 buff\_size

```
int pull_data_s::buff_size
```

recieve buffer size

#### 5.9.2.2 local\_nb\_messages

```
int pull_data_s::local_nb_messages
```

local number of messages

### 5.9.2.3 message\_sizes

```
int* pull_data_s::message_sizes
```

messages sizes, size total\_nb\_messages

### 5.9.2.4 pull\_rank

```
int* pull_data_s::pull_rank
```

array of receiver ranks, size total\_nb\_messages

### 5.9.2.5 push\_rank

```
int* pull_data_s::push_rank
```

array of sender ranks, size total\_nb\_messages

### 5.9.2.6 total\_nb\_messages

```
int pull_data_s::total_nb_messages
```

total number of messages

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

- [/home/tterraz/avido/melissa/Melissa/source/server/server.h](#)

## 5.10 quantile\_s Struct Reference

```
#include <quantile.h>
```

### Public Attributes

- double \* [quantile](#)
- int [increment](#)
- double [alpha](#)

### 5.10.1 Detailed Description

Structure containing an array of quantiles, the corresponding increment, alpha (the quantile partition) and gamma parameters.

## 5.10.2 Member Data Documentation

### 5.10.2.1 alpha

```
double quantile_s::alpha
```

alpha

### 5.10.2.2 increment

```
int quantile_s::increment
```

increment of this quantile

### 5.10.2.3 quantile

```
double* quantile_s::quantile
```

quantile[vect\_size]

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

- </home/tterraz/avido/melissa/Melissa/source/stats/quantile.h>

## 5.11 sobol\_array\_s Struct Reference

```
#include <sobol.h>
```

Collaboration diagram for sobol\_array\_s:

### Public Attributes

- [sobol\\_jansen\\_t](#) \* [sobol\\_jansen](#)
- [sobol\\_martinez\\_t](#) \* [sobol\\_martinez](#)
- [variance\\_t](#) [variance\\_a](#)
- [variance\\_t](#) [variance\\_b](#)
- [int](#) [iteration](#)

### 5.11.1 Detailed Description

Structure containing an array of sobol index structures

### 5.11.2 Member Data Documentation

#### 5.11.2.1 iteration

`int sobol_array_s::iteration`

number of computed groups

#### 5.11.2.2 sobol\_jansen

`sobol_jansen_t* sobol_array_s::sobol_jansen`

array of sobol indices, size nb\_parameters

#### 5.11.2.3 sobol\_martinez

`sobol_martinez_t* sobol_array_s::sobol_martinez`

array of sobol indices, size nb\_parameters

#### 5.11.2.4 variance\_a

`variance_t sobol_array_s::variance_a`

first set variance needed by Martinez formula

#### 5.11.2.5 variance\_b

`variance_t sobol_array_s::variance_b`

second set variance needed by Martinez formula

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

- `/home/tterraz/avido/melissa/Melissa/source/stats/sobol.h`

## 5.12 sobol\_jansen\_s Struct Reference

```
#include <sobol.h>
```

## Public Attributes

- double \* [summ\\_a](#)
- double \* [summ\\_b](#)
- double \* [first\\_order\\_values](#)
- double \* [total\\_order\\_values](#)

### 5.12.1 Detailed Description

Structure containing a sobol indices vector with all structures needed by Jansen update formula

### 5.12.2 Member Data Documentation

#### 5.12.2.1 first\_order\_values

```
double* sobol_jansen_s::first_order_values
```

values of the sobol indices

#### 5.12.2.2 summ\_a

```
double* sobol_jansen_s::summ_a
```

summ needed by Jansen formula

#### 5.12.2.3 summ\_b

```
double* sobol_jansen_s::summ_b
```

summ needed by Jansen formula

#### 5.12.2.4 total\_order\_values

```
double* sobol_jansen_s::total_order_values
```

values of the sobol indices

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

- [/home/tterraz/avido/melissa/Melissa/source/stats/sobol.h](#)



## 5.13 sobol\_martinez\_s Struct Reference

```
#include <sobol.h>
```

Collaboration diagram for sobol\_martinez\_s:

### Public Attributes

- [covariance\\_t](#) first\_order\_covariance
- [covariance\\_t](#) total\_order\_covariance
- [variance\\_t](#) variance\_k
- double \* [first\\_order\\_values](#)
- double \* [total\\_order\\_values](#)
- double [confidence\\_interval](#) [2]

### 5.13.1 Detailed Description

Structure containing a sobol indices vector with all structures needed by Martinez update formula

### 5.13.2 Member Data Documentation

#### 5.13.2.1 confidence\_interval

```
double sobol_martinez_s::confidence_interval[2]
```

interval for 95% confidence level

#### 5.13.2.2 first\_order\_covariance

```
covariance\_t sobol_martinez_s::first_order_covariance
```

covariance needed by Martinez formula

#### 5.13.2.3 first\_order\_values

```
double* sobol_martinez_s::first_order_values
```

values of the sobol indices

#### 5.13.2.4 total\_order\_covariance

```
covariance\_t sobol_martinez_s::total_order_covariance
```

covariance needed by Martinez formula

#### 5.13.2.5 total\_order\_values

`double* sobol_martinez_s::total_order_values`

values of the sobol indices

#### 5.13.2.6 variance\_k

`variance_t sobol_martinez_s::variance_k`

variance needed by Martinez formula

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

- `/home/tterraz/avido/melissa/Melissa/source/stats/sobol.h`

### 5.14 variance\_s Struct Reference

```
#include <variance.h>
```

Collaboration diagram for `variance_s`:

#### Public Attributes

- `double *` `variance`
- `mean_t` `mean_structure`

#### 5.14.1 Detailed Description

Structure containing an array of variances and the corresponding mean structure

#### 5.14.2 Member Data Documentation

##### 5.14.2.1 mean\_structure

`mean_t` `variance_s::mean_structure`

corresponding mean

## 5.14.2.2 variance

```
double* variance_s::variance
```

```
variance[vect_size]
```

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

- /home/tterraz/avido/melissa/Melissa/source/stats/[variance.h](#)

## 5.15 zmq\_data\_s Struct Reference

## Public Attributes

- void \* [context](#)
- void \* [connexion\\_requester](#)
- void \* [init\\_requester](#)
- void \*\* [data\\_pusher](#)
- void \*\* [sobol\\_requester](#)
- int [rinit\\_tab](#) [3]
- int [sobol](#)
- int [sobol\\_rank](#)
- int [sinit\\_tab](#) [2]
- int [nb\\_proc\\_server](#)
- int [nb\\_parameters](#)
- int \* [server\\_vect\\_size](#)
- char \* [buffer](#)
- int [buff\\_size](#)
- int [send\\_buff\\_size](#)
- double \* [buffer\\_sobol](#)
- int \* [send\\_counts](#)
- int \* [local\\_vect\\_sizes](#)
- int \* [sdispls](#)
- int \* [pull\\_rank](#)
- int \* [push\\_rank](#)
- int \* [message\\_sizes](#)
- int [total\\_nb\\_messages](#)
- int [local\\_nb\\_messages](#)
- int [coupling](#)
- [MPI\\_Comm](#) [comm\\_sobol](#)

## 5.15.1 Detailed Description

Structure containing some data needed by zmq

## 5.15.2 Member Data Documentation

#### 5.15.2.1 `buff_size`

```
int zmq_data_s::buff_size
```

size of this buffer

#### 5.15.2.2 `buffer`

```
char* zmq_data_s::buffer
```

buffer used to send data to the library

#### 5.15.2.3 `buffer_sobol`

```
double* zmq_data_s::buffer_sobol
```

buffer used to store data on sobol rank 0

#### 5.15.2.4 `comm_sobol`

```
MPI_Comm zmq_data_s::comm_sobol
```

inter-groups communicator

#### 5.15.2.5 `connexion_requester`

```
void* zmq_data_s::connexion_requester
```

connexion ZeroMQ port

#### 5.15.2.6 `context`

```
void* zmq_data_s::context
```

ZeroMQ context

#### 5.15.2.7 `coupling`

```
int zmq_data_s::coupling
```

coupled simulations or not

#### 5.15.2.8 `data_pusher`

```
void** zmq_data_s::data_pusher
```

push data ZeroMQ ports

#### 5.15.2.9 init\_requester

```
void* zmq_data_s::init_requester
```

initialization ZeroMQ port

#### 5.15.2.10 local\_nb\_messages

```
int zmq_data_s::local_nb_messages
```

local number of messages

#### 5.15.2.11 local\_vect\_sizes

```
int* zmq_data_s::local_vect_sizes
```

local vector size

#### 5.15.2.12 message\_sizes

```
int* zmq_data_s::message_sizes
```

size of the message i

#### 5.15.2.13 nb\_parameters

```
int zmq_data_s::nb_parameters
```

number of parameters of the study

#### 5.15.2.14 nb\_proc\_server

```
int zmq_data_s::nb_proc_server
```

number of MPI processes of the library

#### 5.15.2.15 pull\_rank

```
int* zmq_data_s::pull_rank
```

rank of the pulling process for the message i

#### 5.15.2.16 push\_rank

```
int* zmq_data_s::push_rank
```

rank of the pushing process for the message i

#### 5.15.2.17 rinit\_tab

```
int zmq_data_s::rinit_tab[3]
```

array used to receive data

#### 5.15.2.18 sdispls

```
int* zmq_data_s::sdispls
```

displacement to which data should be sent to server rank i

#### 5.15.2.19 send\_buff\_size

```
int zmq_data_s::send_buff_size
```

size of send buffer

#### 5.15.2.20 send\_counts

```
int* zmq_data_s::send_counts
```

number of elements to send to server rank i

#### 5.15.2.21 server\_vect\_size

```
int* zmq_data_s::server_vect_size
```

local vect size for the library

#### 5.15.2.22 sinit\_tab

```
int zmq_data_s::sinit_tab[2]
```

array used to send data

#### 5.15.2.23 sobol

```
int zmq_data_s::sobol
```

1 if sobol computation, 0 otherwise

#### 5.15.2.24 sobol\_rank

```
int zmq_data_s::sobol_rank
```

sobol rank

#### 5.15.2.25 sobol\_requester

`void** zmq_data_s::sobol_requester`

data ZeroMQ Sobol port

#### 5.15.2.26 total\_nb\_messages

`int zmq_data_s::total_nb_messages`

total number of messages

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

- `/home/tterraz/avido/melissa/Melissa/source/api/melissa\_api.c`





## Chapter 6

# File Documentation

### 6.1 /home/tterraz/avido/melissa/Melissa/source/api/melissa\_api.c File Reference

API Functions.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <zmq.h>
#include "melissa_api_no_mpi.h"
#include "melissa_utils.h"
Include dependency graph for melissa_api.c:
```

### 6.2 /home/tterraz/avido/melissa/Melissa/source/api/melissa\_api.h File Reference

#### 6.2.1 Detailed Description

Author

Terraz Théophile

Date

2016-09-05

### 6.3 /home/tterraz/avido/melissa/Melissa/source/api/melissa\_api\_no\_mpi.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- void [melissa\\_init\\_no\\_mpi](#) (const int \*vect\_size, const int \*sobol\_rank, const int \*sample\_id)
- void [melissa\\_send\\_no\\_mpi](#) (const int \*time\_step, const char \*field\_name, double \*send\_vect, const int \*sobol\_rank, const int \*sample\_id)
- void [melissa\\_finalize](#) ()

### 6.3.1 Detailed Description

**Author**

Terraz Théophile

**Date**

2016-09-05

## 6.4 /home/tterraz/avido/melissa/Melissa/source/server/compute\_stats.c File Reference

Functions called by the server.

```
#include <stdio.h>
#include <stdlib.h>
#include "melissa_data.h"
#include "melissa_utils.h"
Include dependency graph for compute_stats.c:
```

### Functions

- void [compute\\_stats](#) ([melissa\\_data\\_t](#) \*data, const int time\_step, const int nb\_vect, double \*\*in\_vect\_tab, const int group\_id)
- void [finalize\\_stats](#) ([melissa\\_data\\_t](#) \*data)

### 6.4.1 Detailed Description

Functions called by the server.

**Author**

Terraz Théophile

**Date**

2016-15-03

## 6.5 /home/tterraz/avido/melissa/Melissa/source/server/compute\_stats.h File Reference

```
#include "melissa_data.h"
Include dependency graph for compute_stats.h: This graph shows which files directly or indirectly include this file:
```

### Functions

- void [compute\\_stats](#) ([melissa\\_data\\_t](#) \*data, const int time\_step, const int nb\_vect, double \*\*in\_vect\_tab, const int group\_id)
- void [finalize\\_stats](#) ([melissa\\_data\\_t](#) \*data)

### 6.5.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-15-01

## 6.6 /home/tterraz/avido/melissa/Melissa/source/server/fault\_tolerance.c File Reference

```
#include <string.h>
#include "fault_tolerance.h"
#include "zmq.h"
Include dependency graph for fault_tolerance.c:
```

### Functions

- [melissa\\_simulation\\_t](#) \* **add\_simulation** (int id, int nb\_time\_steps)
- void **simu\_push\_to** (vector\_t \*v, int pos, int nb\_time\_steps)
- void **free\_simu\_vector** (vector\_t v)
- int **check\_timeouts** (int \*simu\_state, int \*simu\_timeouts, double \*last\_message\_simu, int nb\_simu)
- void **send\_timeouts** (int detected\_timeouts, int \*simu\_timeouts, int nb\_simu, char \*txt\_buffer, void \*python\_pusher)

### 6.6.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-30-06

## 6.7 /home/tterraz/avido/melissa/Melissa/source/server/fault\_tolerance.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <melissa_utils.h>
#include <vector.h>
Include dependency graph for fault_tolerance.h: This graph shows which files directly or indirectly include this file:
```

### Classes

- struct [melissa\\_simulation\\_s](#)

## Typedefs

- typedef struct field\_status\_s **field\_status\_t**
- typedef struct [melissa\\_simulation\\_s](#) **melissa\_simulation\_t**

## Functions

- [melissa\\_simulation\\_t](#) \* **add\_simulation** (int id, int nb\_time\_steps)
- void **simu\_push\_to** (vector\_t \*v, int pos, int nb\_time\_steps)
- void **free\_simu\_vector** (vector\_t v)
- int **check\_timeouts** (int \*simu\_state, int \*simu\_timeouts, double \*last\_message\_simu, int nb\_simu)
- void **send\_timeouts** (int detected\_timeouts, int \*simu\_timeouts, int nb\_simu, char \*txt\_buffer, void \*python\_requester)

### 6.7.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-30-06

## 6.8 /home/tterraz/avido/melissa/Melissa/source/server/melissa\_data.c File Reference

Routines related to the melissa\_data structure.

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "melissa_data.h"
#include "melissa_utils.h"
#include "sobol.h"
```

Include dependency graph for melissa\_data.c:

## Functions

- void [melissa\\_init\\_data](#) ([melissa\\_data\\_t](#) \*data, [melissa\\_options\\_t](#) \*options, int vect\_size)
- void [melissa\\_check\\_data](#) ([melissa\\_data\\_t](#) \*data)
- void [melissa\\_free\\_data](#) ([melissa\\_data\\_t](#) \*data)
- long int [mem\\_conso](#) ([melissa\\_options\\_t](#) \*options)

### 6.8.1 Detailed Description

Routines related to the melissa\_data structure.

#### Author

Terraz Théophile

#### Date

2016-24-05

### 6.8.2 Function Documentation

#### 6.8.2.1 melissa\_check\_data()

```
void melissa_check_data (  
    melissa_data_t * data )
```

This function checks the data structure. It terminates the program if one of the mandatory option is missing or invalid, and correct mistakes for other options

#### Parameters

in, out	*data	pointer to the structure containing global parameters
---------	-------	---

#### 6.8.2.2 mem\_conso()

```
long int mem_conso (  
    melissa_options_t * options )
```

This function computes and displays the memory consumption of the library

#### Parameters

in	*options	pointer to the structure containing global options
----	----------	--

## 6.9 /home/tterraz/avido/melissa/Melissa/source/server/melissa\_data.h File Reference

```
#include <stdint.h>  
#include "melissa_utils.h"
```

```
#include "melissa_options.h"
#include "mean.h"
#include "variance.h"
#include "min_max.h"
#include "threshold.h"
#include "quantile.h"
#include "covariance.h"
#include "sobol.h"
#include "vector.h"
```

Include dependency graph for melissa\_data.h: This graph shows which files directly or indirectly include this file:

## Classes

- struct [comm\\_data\\_s](#)
- struct [melissa\\_data\\_s](#)

## Typedefs

- typedef struct [comm\\_data\\_s](#) [comm\\_data\\_t](#)
- typedef struct [melissa\\_data\\_s](#) [melissa\\_data\\_t](#)

## Functions

- void [melissa\\_init\\_data](#) ([melissa\\_data\\_t](#) \*data, [melissa\\_options\\_t](#) \*options, int vect\_size)
- void [melissa\\_check\\_data](#) ([melissa\\_data\\_t](#) \*data)
- void [melissa\\_free\\_data](#) ([melissa\\_data\\_t](#) \*data)
- long int [mem\\_conso](#) ([melissa\\_options\\_t](#) \*options)

## 6.9.1 Detailed Description

### Author

Terraz Théophile

### Date

2017-15-01

## 6.9.2 Typedef Documentation

### 6.9.2.1 [comm\\_data\\_t](#)

typedef struct [comm\\_data\\_s](#) [comm\\_data\\_t](#)

type corresponding to [comm\\_data\\_s](#)

### 6.9.2.2 melissa\_data\_t

```
typedef struct melissa_data_s melissa_data_t
```

type corresponding to [melissa\\_data\\_s](#)

## 6.9.3 Function Documentation

### 6.9.3.1 melissa\_check\_data()

```
void melissa_check_data (
    melissa_data_t * data )
```

This function checks the data structure. It terminates the program if one of the mandatory option is missing or invalid, and correct mistakes for other options

#### Parameters

in, out	*data	pointer to the structure containing global parameters
---------	-------	---

### 6.9.3.2 mem\_conso()

```
long int mem_conso (
    melissa_options_t * options )
```

This function computes and displays the memory consumption of the library

#### Parameters

in	*options	pointer to the structure containing global options
----	----------	--

## 6.10 /home/tterraz/avido/melissa/Melissa/source/server/melissa\_fields.c File Reference

Routines related to the melissa\_fields structure.

```
#include <getopt.h>
#include <string.h>
#include "melissa_fields.h"
#include "melissa_data.h"
#include "melissa_io.h"
#include "compute_stats.h"
Include dependency graph for melissa_fields.c:
```

## Functions

- void **melissa\_get\_fields** (int argc, char \*\*argv, [melissa\\_field\\_t](#) \*fields, int nb\_fields)
- void **add\_fields** ([melissa\\_field\\_t](#) \*field, int data\_size, int nb\_fields)
- [melissa\\_data\\_t](#) \* **get\_data\_ptr** ([melissa\\_field\\_t](#) \*field, int nb\_fields, char \*field\_name)
- void **finalize\_field\_data** ([melissa\\_field\\_t](#) \*field, [comm\\_data\\_t](#) \*comm\_data, [melissa\\_options\\_t](#) \*options, int \*local\_vect\_sizes)

### 6.10.1 Detailed Description

Routines related to the melissa\_fields structure.

#### Author

Terraz Théophile

#### Date

2017-01-09

## 6.11 /home/terraz/avido/melissa/Melissa/source/server/melissa\_fields.h File Reference

```
#include "melissa_data.h"
```

Include dependency graph for melissa\_fields.h: This graph shows which files directly or indirectly include this file:

## Classes

- struct [melissa\\_field\\_s](#)

## Typedefs

- typedef struct [melissa\\_field\\_s](#) [melissa\\_field\\_t](#)

## Functions

- void **melissa\_get\_fields** (int argc, char \*\*argv, [melissa\\_field\\_t](#) \*fields, int nb\_fields)
- void **add\_fields** ([melissa\\_field\\_t](#) \*field, int data\_size, int nb\_fields)
- [melissa\\_data\\_t](#) \* **get\_data\_ptr** ([melissa\\_field\\_t](#) \*field, int nb\_fields, char \*field\_name)
- void **finalize\_field\_data** ([melissa\\_field\\_t](#) \*field, [comm\\_data\\_t](#) \*comm\_data, [melissa\\_options\\_t](#) \*options, int \*local\_vect\_sizes)

### 6.11.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-15-01



## 6.11.2 Typedef Documentation

### 6.11.2.1 melissa\_field\_t

```
typedef struct melissa_field_s melissa_field_t
```

type corresponding to field\_s

## 6.12 /home/tterraz/avido/melissa/Melissa/source/server/melissa\_io.c File Reference

Inputs, outputs and checkpoints.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <stdint.h>
#include <math.h>
#include "melissa_data.h"
#include "melissa_utils.h"
Include dependency graph for melissa_io.c:
```

## Functions

- void [write\\_client\\_data](#) (int \*client\_comm\_size, int \*client\_vect\_sizes)
- int [read\\_client\\_data](#) (int \*client\_comm\_size, int \*\*client\_vect\_sizes, [melissa\\_options\\_t](#) \*options)
- void [save\\_stats](#) ([melissa\\_data\\_t](#) \*data, [comm\\_data\\_t](#) \*comm\_data, char \*field\_name)
- void [read\\_saved\\_stats](#) ([melissa\\_data\\_t](#) \*data, [comm\\_data\\_t](#) \*comm\_data, char \*field\_name, int client\_rank)
- void [save\\_simu\\_states](#) (int \*simu\_states, [comm\\_data\\_t](#) \*comm\_data, int size)
- void [read\\_simu\\_states](#) (int \*simu\_states, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int size)
- void [write\\_stats\\_bin](#) ([melissa\\_data\\_t](#) \*\*data, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int \*local\_vect\_sizes, char \*field)
- void [write\\_stats\\_ensight](#) ([melissa\\_data\\_t](#) \*\*data, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int \*local\_vect\_sizes, char \*field)
- void [read\\_ensight](#) ([melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, double \*in\_vect, int \*local\_vect\_sizes, char \*file\_name)
- void [write\\_stats\\_txt](#) ([melissa\\_data\\_t](#) \*\*data, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int \*local\_vect\_sizes, char \*field)

### 6.12.1 Detailed Description

Inputs, outputs and checkpoints.

#### Author

Terraz Théophile

#### Date

2017-19-01

## 6.13 /home/tterraz/avido/melissa/Melissa/source/server/melissa\_io.h File Reference

```
#include "melissa_data.h"
#include "melissa_options.h"
```

Include dependency graph for melissa\_io.h: This graph shows which files directly or indirectly include this file:

### Functions

- void [write\\_stats\\_bin](#) ([melissa\\_data\\_t](#) \*\*data, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int \*local\_vect\_sizes, char \*field)
- void [write\\_stats\\_txt](#) ([melissa\\_data\\_t](#) \*\*data, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int \*local\_vect\_sizes, char \*field)
- void [write\\_stats\\_ensight](#) ([melissa\\_data\\_t](#) \*\*data, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int \*local\_vect\_sizes, char \*field)
- void [write\\_client\\_data](#) (int \*client\_comm\_size, int \*client\_vect\_sizes)
- int [read\\_client\\_data](#) (int \*client\_comm\_size, int \*\*client\_vect\_sizes, [melissa\\_options\\_t](#) \*options)
- void [save\\_stats](#) ([melissa\\_data\\_t](#) \*data, [comm\\_data\\_t](#) \*comm\_data, char \*field\_name)
- void [read\\_saved\\_stats](#) ([melissa\\_data\\_t](#) \*data, [comm\\_data\\_t](#) \*comm\_data, char \*field\_name, int client\_rank)
- void [save\\_simu\\_states](#) (int \*simu\_states, [comm\\_data\\_t](#) \*comm\_data, int size)
- void [read\\_simu\\_states](#) (int \*simu\_states, [melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, int size)
- void [read\\_ensight](#) ([melissa\\_options\\_t](#) \*options, [comm\\_data\\_t](#) \*comm\_data, double \*in\_vect, int \*local\_vect\_sizes, char \*file\_name)

### 6.13.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-15-01

## 6.14 /home/tterraz/avido/melissa/Melissa/source/server/melissa\_options.c File Reference

Parse commande line to get stats options.

```
#include <unistd.h>
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include "melissa_options.h"
#include "melissa_data.h"
#include "melissa_utils.h"
Include dependency graph for melissa_options.c:
```

## Functions

- void **get\_fields** (char \*name, [melissa\\_options\\_t](#) \*options)
- void [melissa\\_print\\_options](#) ([melissa\\_options\\_t](#) \*options)
- void [melissa\\_get\\_options](#) (int argc, char \*\*argv, [melissa\\_options\\_t](#) \*options)
- void [melissa\\_check\\_options](#) ([melissa\\_options\\_t](#) \*options)
- void [melissa\\_write\\_options](#) ([melissa\\_options\\_t](#) \*options)
- int [melissa\\_read\\_options](#) ([melissa\\_options\\_t](#) \*options)

### 6.14.1 Detailed Description

Parse commande line to get stats options.

#### Author

Terraz Théophile

#### Date

2016-03-03

## 6.15 /home/tterraz/avido/melissa/Melissa/source/server/melissa\_options.h File Reference

This graph shows which files directly or indirectly include this file:

## Classes

- struct [melissa\\_options\\_s](#)

## Typedefs

- typedef struct [melissa\\_options\\_s](#) [melissa\\_options\\_t](#)

## Functions

- void [melissa\\_get\\_options](#) (int argc, char \*\*argv, [melissa\\_options\\_t](#) \*options)
- void [melissa\\_check\\_options](#) ([melissa\\_options\\_t](#) \*options)
- void [melissa\\_print\\_options](#) ([melissa\\_options\\_t](#) \*options)
- void [melissa\\_write\\_options](#) ([melissa\\_options\\_t](#) \*options)
- int [melissa\\_read\\_options](#) ([melissa\\_options\\_t](#) \*options)

### 6.15.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-15-01

### 6.15.2 Typedef Documentation

#### 6.15.2.1 melissa\_options\_t

```
typedef struct melissa_options_s melissa_options_t
```

type corresponding to [melissa\\_options\\_s](#)

## 6.16 /home/tterraz/avido/melissa/Melissa/source/server/server.h File Reference

```
#include "melissa_fields.h"
#include "melissa_data.h"
#include "melissa_utils.h"
Include dependency graph for server.h:
```

### Classes

- struct [pull\\_data\\_s](#)

### Typedefs

- typedef struct [pull\\_data\\_s](#) [pull\\_data\\_t](#)

### Functions

- void **comm\_n\_to\_m\_init** (int \*rcounts, int \*rdispls, const int global\_vect\_size, const int \*server\_vect\_size, int \*client\_vect\_size, const int nb\_proc\_client, const int rank, [pull\\_data\\_t](#) \*pull\_data)
- void **increment\_step\_simu** ([melissa\\_field\\_t](#) \*field, char \*field\_name, int group\_id)
- int **check\_simu\_state** ([melissa\\_field\\_t](#) \*field, int nb\_fields, int group\_id, int nb\_time\_steps, [comm\\_data\\_t](#) \*comm\_data)
- long int **count\_mbytes\_written** ([melissa\\_options\\_t](#) \*options)
- int **string\_recv** (void \*socket, char \*recv\_buff)
- void **global\_confidence\_sobol\_martinez** ([melissa\\_field\\_t](#) \*field, int nb\_fields, [comm\\_data\\_t](#) \*comm\_data, double \*interval1, double \*interval\_tot)

## 6.16.1 Detailed Description

### Author

Terraz Théophile

### Date

2016-15-03

## 6.16.2 Typedef Documentation

### 6.16.2.1 pull\_data\_t

```
typedef struct pull_data_s pull_data_t
```

type corresponding to [pull\\_data\\_s](#)

## 6.16.3 Function Documentation

### 6.16.3.1 global\_confidence\_sobol\_martinez()

```
void global_confidence_sobol_martinez (
    melissa_field_t * field,
    int nb_fields,
    comm_data_t * comm_data,
    double * intervall,
    double * interval_tot )
```

This function computes the confidence interval for Martinez Sobol indices

#### Parameters

in	<i>field</i>	array of field structures
in	<i>nb_fields</i>	size of field array
out	<i>*comm_data</i>	comm data structure
out	<i>*intervall</i>	worst confidence interval (first order)
out	<i>*interval_tot</i>	worst confidence interval (total order)

## 6.17 /home/tterraz/avido/melissa/Melissa/source/stats/covariance.c File Reference

Functions needed to compute covariances.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "mean.h"
#include "variance.h"
#include "covariance.h"
#include "melissa_utils.h"
Include dependency graph for covariance.c:
```

### Functions

- void [init\\_covariance](#) ([covariance\\_t](#) \*covariance, const int vect\_size)
- void [increment\\_covariance](#) ([covariance\\_t](#) \*covariance, double in\_vect1[], double in\_vect2[], const int vect\_size)
- void [update\\_covariance](#) ([covariance\\_t](#) \*covariance1, [covariance\\_t](#) \*covariance2, [covariance\\_t](#) \*updated\_covariance, const int vect\_size)
- void [save\\_covariance](#) ([covariance\\_t](#) \*covars, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_covariance](#) ([covariance\\_t](#) \*covars, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [free\\_covariance](#) ([covariance\\_t](#) \*covariance)

### 6.17.1 Detailed Description

Functions needed to compute covariances.

#### Author

Terraz Théophile

#### Date

2016-01-07

### 6.17.2 Function Documentation

#### 6.17.2.1 [free\\_covariance\(\)](#)

```
void free_covariance (
    covariance\_t * covariance )
```

This function frees a covariance structure.

## Parameters

in	<i>*covariance</i>	the covariance structure to free
----	--------------------	----------------------------------

## 6.17.2.2 increment\_covariance()

```
void increment_covariance (
    covariance_t * covariance,
    double in_vect1[],
    double in_vect2[],
    const int vect_size )
```

This function updates the incremental covariance.

## Parameters

in, out	<i>*covariance</i>	input: previously computed covariance, output: incremented covariance
in	<i>in_vect1[]</i>	first input vector of double values
in	<i>in_vect2[]</i>	second input vector of double values
in	<i>vect_size</i>	size of the input vectors

## 6.17.2.3 init\_covariance()

```
void init_covariance (
    covariance_t * covariance,
    const int vect_size )
```

This function initializes a covariance structure.

## Parameters

in, out	<i>*covariance</i>	the covariance structure to initialize
in	<i>vect_size</i>	size of the covariance vector

## 6.17.2.4 read\_covariance()

```
void read_covariance (
    covariance_t * covars,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of covariances structures on disc

**Parameters**

in	<i>*covars</i>	covariance structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.17.2.5 save\_covariance()**

```
void save_covariance (
    covariance_t * covars,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function writes an array of covariances structures on disc

**Parameters**

in	<i>*covars</i>	covariance structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.17.2.6 update\_covariance()**

```
void update_covariance (
    covariance_t * covariance1,
    covariance_t * covariance2,
    covariance_t * updated_covariance,
    const int vect_size )
```

This function updates the incremental covariance.

**Parameters**

in	<i>*covariance1</i>	first input partial covariance
in	<i>*covariance2</i>	second input partial covariance
out	<i>*updated_covariance</i>	updated covariance
in	<i>vect_size</i>	size of the input vectors



## 6.18 /home/tterraz/avido/melissa/Melissa/source/stats/covariance.h File Reference

```
#include "mean.h"
```

Include dependency graph for covariance.h: This graph shows which files directly or indirectly include this file:

### Classes

- struct [covariance\\_s](#)

### Typedefs

- typedef struct [covariance\\_s](#) [covariance\\_t](#)

### Functions

- void [init\\_covariance](#) ([covariance\\_t](#) \*covariance, const int vect\_size)
- void [increment\\_covariance](#) ([covariance\\_t](#) \*partial\_covariance, double in\_vect1[], double in\_vect2[], const int vect\_size)
- void [update\\_covariance](#) ([covariance\\_t](#) \*covariance1, [covariance\\_t](#) \*covariance2, [covariance\\_t](#) \*updated\_↔ covariance, const int vect\_size)
- void [save\\_covariance](#) ([covariance\\_t](#) \*covars, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_covariance](#) ([covariance\\_t](#) \*covars, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [free\\_covariance](#) ([covariance\\_t](#) \*covariance)

### 6.18.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-15-01

### 6.18.2 Typedef Documentation

#### 6.18.2.1 [covariance\\_t](#)

```
typedef struct covariance\_s covariance\_t
```

type corresponding to [covariance\\_s](#)

### 6.18.3 Function Documentation

#### 6.18.3.1 [free\\_covariance\(\)](#)

```
void free_covariance (  
    covariance\_t * covariance )
```

This function frees a covariance structure.

## Parameters

in	<i>*covariance</i>	the covariance structure to free
----	--------------------	----------------------------------

## 6.18.3.2 increment\_covariance()

```
void increment_covariance (
    covariance_t * covariance,
    double in_vect1[],
    double in_vect2[],
    const int vect_size )
```

This function updates the incremental covariance.

## Parameters

in, out	<i>*covariance</i>	input: previously computed covariance, output: incremented covariance
in	<i>in_vect1[]</i>	first input vector of double values
in	<i>in_vect2[]</i>	second input vector of double values
in	<i>vect_size</i>	size of the input vectors

## 6.18.3.3 init\_covariance()

```
void init_covariance (
    covariance_t * covariance,
    const int vect_size )
```

This function initializes a covariance structure.

## Parameters

in, out	<i>*covariance</i>	the covariance structure to initialize
in	<i>vect_size</i>	size of the covariance vector

## 6.18.3.4 read\_covariance()

```
void read_covariance (
    covariance_t * covars,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of covariances structures on disc

**Parameters**

in	<i>*covars</i>	covariance structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.18.3.5 save\_covariance()**

```
void save_covariance (
    covariance_t * covars,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function writes an array of covariances structures on disc

**Parameters**

in	<i>*covars</i>	covariance structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.18.3.6 update\_covariance()**

```
void update_covariance (
    covariance_t * covariance1,
    covariance_t * covariance2,
    covariance_t * updated_covariance,
    const int vect_size )
```

This function updates the incremental covariance.

**Parameters**

in	<i>*covariance1</i>	first input partial covariance
in	<i>*covariance2</i>	second input partial covariance
out	<i>*updated_covariance</i>	updated covariance
in	<i>vect_size</i>	size of the input vectors

## 6.19 /home/tterraz/avido/melissa/Melissa/source/stats/mean.c File Reference

Mean related functions.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include "mean.h"
#include "melissa_utils.h"
Include dependency graph for mean.c:
```

### Functions

- void [init\\_mean](#) ([mean\\_t](#) \*mean, const int vect\_size)
- void [increment\\_mean](#) ([mean\\_t](#) \*mean, double in\_vect[], const int vect\_size)
- void [update\\_mean](#) ([mean\\_t](#) \*mean1, [mean\\_t](#) \*mean2, [mean\\_t](#) \*updated\_mean, const int vect\_size)
- void [save\\_mean](#) ([mean\\_t](#) \*means, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_mean](#) ([mean\\_t](#) \*means, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [free\\_mean](#) ([mean\\_t](#) \*mean)

### 6.19.1 Detailed Description

Mean related functions.

#### Author

Terraz Théophile

#### Date

2016-15-02

### 6.19.2 Function Documentation

#### 6.19.2.1 [free\\_mean\(\)](#)

```
void free_mean (
    mean\_t * mean )
```

This function frees a mean structure.

#### Parameters

in	* <i>mean</i>	the mean structure to free
----	---------------	----------------------------

### 6.19.2.2 increment\_mean()

```
void increment_mean (
    mean_t * mean,
    double in_vect[],
    const int vect_size )
```

This function updates the incremental mean.

#### Parameters

in, out	<i>*mean</i>	input: previously computed iterative mean, output: updated mean
in	<i>in_vect[]</i>	input vector of double values
in	<i>vect_size</i>	size of the input vectors

### 6.19.2.3 init\_mean()

```
void init_mean (
    mean_t * mean,
    const int vect_size )
```

This function initializes a mean structure.

#### Parameters

in, out	<i>*mean</i>	the mean structure to initialize
in	<i>vect_size</i>	size of the mean vector

### 6.19.2.4 read\_mean()

```
void read_mean (
    mean_t * means,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of mean structures on disc

#### Parameters

in	<i>*means</i>	mean structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

#### 6.19.2.5 save\_mean()

```
void save_mean (
    mean_t * means,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function writes an array of mean structures on disc

##### Parameters

in	<i>*means</i>	mean structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

#### 6.19.2.6 update\_mean()

```
void update_mean (
    mean_t * mean1,
    mean_t * mean2,
    mean_t * updated_mean,
    const int vect_size )
```

This function aggregates two partial means.

##### Parameters

in	<i>*mean1</i>	first input vector of partial means
in	<i>*mean2</i>	second input vector of partial means
out	<i>*updated_mean</i>	the updated mean
in	<i>vect_size</i>	size of the input and output vectors

## 6.20 /home/tterraz/avido/melissa/Melissa/source/stats/mean.h File Reference

```
#include <stdio.h>
```

Include dependency graph for mean.h: This graph shows which files directly or indirectly include this file:

### Classes

- struct [mean\\_s](#)

## Typedefs

- typedef struct [mean\\_s](#) [mean\\_t](#)

## Functions

- void [init\\_mean](#) ([mean\\_t](#) \*mean, const int vect\_size)
- void [increment\\_mean](#) ([mean\\_t](#) \*mean, double in\_vect[], const int vect\_size)
- void [update\\_mean](#) ([mean\\_t](#) \*mean1, [mean\\_t](#) \*mean2, [mean\\_t](#) \*updated\_mean, const int vect\_size)
- void [save\\_mean](#) ([mean\\_t](#) \*means, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_mean](#) ([mean\\_t](#) \*means, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [free\\_mean](#) ([mean\\_t](#) \*mean)

### 6.20.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-15-01

### 6.20.2 Typedef Documentation

#### 6.20.2.1 [mean\\_t](#)

```
typedef struct mean\_s mean\_t
```

type corresponding to [mean\\_s](#)

### 6.20.3 Function Documentation

#### 6.20.3.1 [free\\_mean\(\)](#)

```
void free_mean (  
    mean\_t * mean )
```

This function frees a mean structure.

## Parameters

in	<i>*mean</i>	the mean structure to free
----	--------------	----------------------------

## 6.20.3.2 increment\_mean()

```
void increment_mean (
    mean_t * mean,
    double in_vect[],
    const int vect_size )
```

This function updates the incremental mean.

## Parameters

in, out	<i>*mean</i>	input: previously computed iterative mean, output: updated mean
in	<i>in_vect[]</i>	input vector of double values
in	<i>vect_size</i>	size of the input vectors

## 6.20.3.3 init\_mean()

```
void init_mean (
    mean_t * mean,
    const int vect_size )
```

This function initializes a mean structure.

## Parameters

in, out	<i>*mean</i>	the mean structure to initialize
in	<i>vect_size</i>	size of the mean vector

## 6.20.3.4 read\_mean()

```
void read_mean (
    mean_t * means,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of mean structures on disc



**Parameters**

in	<i>*means</i>	mean structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.20.3.5 save\_mean()**

```
void save_mean (
    mean_t * means,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function writes an array of mean structures on disc

**Parameters**

in	<i>*means</i>	mean structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.20.3.6 update\_mean()**

```
void update_mean (
    mean_t * mean1,
    mean_t * mean2,
    mean_t * updated_mean,
    const int vect_size )
```

This function aggregates two partial means.

**Parameters**

in	<i>*mean1</i>	first input vector of partial means
in	<i>*mean2</i>	second input vector of partial means
out	<i>*updated_mean</i>	the updated mean
in	<i>vect_size</i>	size of the input and output vectors

## 6.21 /home/terraz/avido/melissa/Melissa/source/stats/min\_max.c File Reference

Min and max related functions.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "min_max.h"
#include "melissa_utils.h"
Include dependency graph for min_max.c:
```

### Functions

- void [init\\_min\\_max](#) ([min\\_max\\_t](#) \*min\_max, const int vect\_size)
- void [min\\_and\\_max](#) ([min\\_max\\_t](#) \*min\_max, double in\_vect[], const int vect\_size)
- void [save\\_min\\_max](#) ([min\\_max\\_t](#) \*minmax, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_min\\_max](#) ([min\\_max\\_t](#) \*minmax, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [free\\_min\\_max](#) ([min\\_max\\_t](#) \*min\_max)

### 6.21.1 Detailed Description

Min and max related functions.

#### Author

Terraz Théophile

#### Date

2016-15-02

### 6.21.2 Function Documentation

#### 6.21.2.1 [free\\_min\\_max\(\)](#)

```
void free_min_max (
    min\_max\_t * min_max )
```

This function frees a min and max structure.

#### Parameters

in	* <a href="#">min_max</a>	the min and max structure to free
----	---------------------------	-----------------------------------

### 6.21.2.2 init\_min\_max()

```
void init_min_max (
    min_max_t * min_max,
    const int vect_size )
```

This function initializes a min and max structure.

#### Parameters

in, out	<i>*min_max</i>	the min and max structure to initialize
in	<i>vect_size</i>	size of the vectors

### 6.21.2.3 min\_and\_max()

```
void min_and_max (
    min_max_t * min_max,
    double in_vect[],
    const int vect_size )
```

This function updates the min and the max values of min and max vectors using the input vector.

#### Parameters

in, out	<i>*min_max</i>	the min and max structure
in	<i>in_vect[]</i>	input vector of double values
in	<i>vect_size</i>	size of the input vectors

### 6.21.2.4 read\_min\_max()

```
void read_min_max (
    min_max_t * minmax,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of min and max structures on disc

#### Parameters

in	<i>*minmax</i>	min and max structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

### 6.21.2.5 save\_min\_max()

```
void save_min_max (
    min_max_t * minmax,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function writes an array of min and max structures on disc

#### Parameters

in	<i>*minmax</i>	min and max structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

## 6.22 /home/terraz/avido/melissa/Melissa/source/stats/min\_max.h File Reference

This graph shows which files directly or indirectly include this file:

### Classes

- struct [min\\_max\\_s](#)

### Typedefs

- typedef struct [min\\_max\\_s](#) [min\\_max\\_t](#)

### Functions

- void [init\\_min\\_max](#) ([min\\_max\\_t](#) \*min\_max, const int vect\_size)
- void [min\\_and\\_max](#) ([min\\_max\\_t](#) \*min\_max, double in\_vect[], const int vect\_size)
- void [save\\_min\\_max](#) ([min\\_max\\_t](#) \*minmax, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_min\\_max](#) ([min\\_max\\_t](#) \*minmax, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [free\\_min\\_max](#) ([min\\_max\\_t](#) \*min\_max)

### 6.22.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-15-01

## 6.22.2 Typedef Documentation

### 6.22.2.1 min\_max\_t

```
typedef struct min_max_s min_max_t
```

type corresponding to `min_max_s`

## 6.22.3 Function Documentation

### 6.22.3.1 free\_min\_max()

```
void free_min_max (  
    min_max_t * min_max )
```

This function frees a min and max structure.

#### Parameters

in	* <i>min_max</i>	the min and max structure to free
----	------------------	-----------------------------------

### 6.22.3.2 init\_min\_max()

```
void init_min_max (  
    min_max_t * min_max,  
    const int vect_size )
```

This function initializes a min and max structure.

#### Parameters

in, out	* <i>min_max</i>	the min and max structure to initialize
in	<i>vect_size</i>	size of the vectors

### 6.22.3.3 min\_and\_max()

```
void min_and_max (  
    min_max_t * min_max,
```

```
double in_vect[],
const int vect_size )
```

This function updates the min and the max values of min and max vectors using the input vector.

#### Parameters

in, out	<i>*min_max</i>	the min and max structure
in	<i>in_vect[]</i>	input vector of double values
in	<i>vect_size</i>	size of the input vectors

#### 6.22.3.4 read\_min\_max()

```
void read_min_max (
    min_max_t * minmax,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of min and max structures on disc

#### Parameters

in	<i>*minmax</i>	min and max structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

#### 6.22.3.5 save\_min\_max()

```
void save_min_max (
    min_max_t * minmax,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function writes an array of min and max structures on disc

#### Parameters

in	<i>*minmax</i>	min and max structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

## 6.23 /home/tterraz/avido/melissa/Melissa/source/stats/quantile.c File Reference

Quantile related functions.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include "quantile.h"
#include "melissa_utils.h"
Include dependency graph for quantile.c:
```

### Functions

- void [init\\_quantile](#) ([quantile\\_t](#) \*quantile, const int vect\_size, const double alpha)
- void [increment\\_quantile](#) ([quantile\\_t](#) \*quantile, const int nmax, double in\_vect[], const int vect\_size)
- void [save\\_quantile](#) ([quantile\\_t](#) \*quantiles, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_quantile](#) ([quantile\\_t](#) \*quantiles, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [free\\_quantile](#) ([quantile\\_t](#) \*quantile)

#### 6.23.1 Detailed Description

Quantile related functions.

##### Author

Terraz Théophile

##### Date

2017-18-05

#### 6.23.2 Function Documentation

##### 6.23.2.1 free\_quantile()

```
void free_quantile (
    quantile\_t * quantile )
```

This function frees a quantile structure.

##### Parameters

<i>in, out</i>	<i>*quantile</i>	the quantile structure to free
----------------	------------------	--------------------------------

### 6.23.2.2 increment\_quantile()

```
void increment_quantile (
    quantile_t * quantile,
    const int nmax,
    double in_vect[],
    const int vect_size )
```

This function updates the incremental quantile.

#### Parameters

in, out	<i>*quantile</i>	input: previously computed iterative quantile, output: updated partial quantile
in	<i>nmax</i>	maximum number of iterations
in	<i>in_vect[]</i>	input vector of double values
in	<i>vect_size</i>	size of the input vectors

### 6.23.2.3 init\_quantile()

```
void init_quantile (
    quantile_t * quantile,
    const int vect_size,
    const double alpha )
```

This function initializes a quantile structure.

#### Parameters

in, out	<i>*quantile</i>	the quantile structure to initialize
in	<i>vect_size</i>	size of the quantile vector
in	<i>alpha</i>	alpha parameter of the algorithm

### 6.23.2.4 read\_quantile()

```
void read_quantile (
    quantile_t * quantiles,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of quantile structures on disc



## Parameters

in	<i>*quantiles</i>	quantile structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

## 6.23.2.5 save\_quantile()

```
void save_quantile (
    quantile_t * quantiles,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function writes an array of quantile structures on disc

## Parameters

in	<i>*quantiles</i>	quantile structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

## 6.24 /home/tterraz/avido/melissa/Melissa/source/stats/quantile.h File Reference

```
#include <stdio.h>
```

Include dependency graph for quantile.h: This graph shows which files directly or indirectly include this file:

## Classes

- struct [quantile\\_s](#)

## Typedefs

- typedef struct [quantile\\_s](#) [quantile\\_t](#)

## Functions

- void [init\\_quantile](#) ([quantile\\_t](#) \*quantile, const int vect\_size, const double alpha)
- void [increment\\_quantile](#) ([quantile\\_t](#) \*quantile, const int nmax, double in\_vect[], const int vect\_size)
- void [save\\_quantile](#) ([quantile\\_t](#) \*quantile, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_quantile](#) ([quantile\\_t](#) \*quantile, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [free\\_quantile](#) ([quantile\\_t](#) \*quantile)

### 6.24.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-18-05

### 6.24.2 Typedef Documentation

#### 6.24.2.1 `quantile_t`

```
typedef struct quantile_s quantile_t
```

type corresponding to `quantile_s`

### 6.24.3 Function Documentation

#### 6.24.3.1 `free_quantile()`

```
void free_quantile (
    quantile_t * quantile )
```

This function frees a quantile structure.

##### Parameters

<code>in, out</code>	<code>*quantile</code>	the quantile structure to free
----------------------	------------------------	--------------------------------

#### 6.24.3.2 `increment_quantile()`

```
void increment_quantile (
    quantile_t * quantile,
    const int nmax,
    double in_vect[],
    const int vect_size )
```

This function updates the incremental quantile.

**Parameters**

in, out	<i>*quantile</i>	input: previously computed iterative quantile, output: updated partial quantile
in	<i>nmax</i>	maximum number of iterations
in	<i>in_vect[]</i>	input vector of double values
in	<i>vect_size</i>	size of the input vectors

**6.24.3.3 init\_quantile()**

```
void init_quantile (
    quantile_t * quantile,
    const int vect_size,
    const double alpha )
```

This function initializes a quantile structure.

**Parameters**

in, out	<i>*quantile</i>	the quantile structure to initialize
in	<i>vect_size</i>	size of the quantile vector
in	<i>alpha</i>	alpha parameter of the algorithm

**6.24.3.4 read\_quantile()**

```
void read_quantile (
    quantile_t * quantiles,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of quantile structures on disc

**Parameters**

in	<i>*quantiles</i>	quantile structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.24.3.5 save\_quantile()**

```
void save_quantile (
    quantile_t * quantiles,
```

```

    int vect_size,
    int nb_time_steps,
    FILE * f )

```

This function writes an array of quantile structures on disc

#### Parameters

in	<i>*quantiles</i>	quantile structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

## 6.25 /home/tterraz/avido/melissa/Melissa/source/stats/sobol.c File Reference

Functions needed to compute sobol indices.

```

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include "mean.h"
#include "variance.h"
#include "covariance.h"
#include "sobol.h"
#include "melissa_utils.h"
Include dependency graph for sobol.c:

```

### Functions

- void [init\\_sobol\\_jansen](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters, int vect\_size)
- void [init\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters, int vect\_size)
- void [increment\\_sobol\\_jansen](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters, double \*\*in\_vect\_tab, int vect\_size)
- void [increment\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters, double \*\*in\_vect\_tab, int vect\_size)
- void [confidence\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters, int vect\_size)
- int [check\\_convergence\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*\*sobol\_array, double confidence\_value, int nb\_time\_steps, int nb\_parameters)
- void [save\\_sobol\\_jansen](#) ([sobol\\_array\\_t](#) \*sobol\_array, int vect\_size, int nb\_time\_steps, int nb\_parameters, FILE \*f)
- void [save\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int vect\_size, int nb\_time\_steps, int nb\_parameters, FILE \*f)
- void [read\\_sobol\\_jansen](#) ([sobol\\_array\\_t](#) \*sobol\_array, int vect\_size, int nb\_time\_steps, int nb\_parameters, FILE \*f)
- void [read\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int vect\_size, int nb\_time\_steps, int nb\_parameters, FILE \*f)
- void [free\\_sobol\\_jansen](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters)
- void [free\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters)

### 6.25.1 Detailed Description

Functions needed to compute sobol indices.

#### Author

Terraz Théophile

#### Date

2016-29-02

### 6.25.2 Function Documentation

#### 6.25.2.1 check\_convergence\_sobol\_martinez()

```
int check_convergence_sobol_martinez (
    sobol_array_t ** sobol_array,
    double confidence_value,
    int nb_time_steps,
    int nb_parameters )
```

This function check if the Sobol indice convergence has been reached

#### Parameters

out	<i>**sobol_array</i>	Sobol indices
in	<i>confidence_value</i>	value to reach for the worst confidence interval
in	<i>nb_time_steps</i>	number of time steps of the study
in	<i>nb_parameters</i>	size of sobol_array->sobol_martinez

#### Returns

[out] int 0 if convergence is not reached 1 if convergence is reached

#### 6.25.2.2 confidence\_sobol\_martinez()

```
void confidence_sobol_martinez (
    sobol_array_t * sobol_array,
    int nb_parameters,
    int vect_size )
```

This function computes the confidence interval for Martinez Sobol indices

**Parameters**

out	<i>*sobol_array</i>	Sobol indices
in	<i>nb_parameters</i>	size of sobol_array->sobol_martinez
in	<i>vect_size</i>	size of input vectors

**6.25.2.3 free\_sobol\_jansen()**

```
void free_sobol_jansen (
    sobol_array_t * sobol_array,
    int nb_parameters )
```

This function frees a Jansen Sobol array structure

**Parameters**

in	<i>*sobol_array</i>	reference or pointer to a sobol index structure to free
in	<i>nb_parameters</i>	number of parameters of the study

**6.25.2.4 free\_sobol\_martinez()**

```
void free_sobol_martinez (
    sobol_array_t * sobol_array,
    int nb_parameters )
```

This function frees a Martinez Sobol indices structure

**Parameters**

in	<i>*sobol_array</i>	reference or pointer to a sobol array structure to free
in	<i>nb_parameters</i>	number of parameters of the study

**6.25.2.5 increment\_sobol\_jansen()**

```
void increment_sobol_jansen (
    sobol_array_t * sobol_array,
    int nb_parameters,
    double ** in_vect_tab,
    int vect_size )
```

This function computes Sobol indices using Jansen formula

**Parameters**

out	<i>*sobol_array</i>	computed sobol indices, using Jansen formula
in	<i>nb_parameters</i>	size of sobol_array->sobol_jansen
in	<i>**in_vect_tab</i>	array of input vectors
in	<i>vect_size</i>	size of input vectors

**6.25.2.6 increment\_sobol\_martinez()**

```
void increment_sobol_martinez (
    sobol_array_t * sobol_array,
    int nb_parameters,
    double ** in_vect_tab,
    int vect_size )
```

This function computes Sobol indices using Martinez formula

**Parameters**

out	<i>*sobol_array</i>	computed sobol indices, using Martinez formula
in	<i>nb_parameters</i>	size of sobol_array->sobol_martinez
in	<i>**in_vect_tab</i>	array of input vectors
in	<i>vect_size</i>	size of input vectors

**6.25.2.7 init\_sobol\_jansen()**

```
void init_sobol_jansen (
    sobol_array_t * sobol_array,
    int nb_parameters,
    int vect_size )
```

This function initialise a Jansen Sobol indices structure

**Parameters**

in, out	<i>*sobol_array</i>	input: reference or pointer to an uninitialised sobol indices structure, output: initialised structure, with values and variances set to 0
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>vect_size</i>	size of the input vectors

**6.25.2.8 init\_sobol\_martinez()**

```
void init_sobol_martinez (
```

```

sobol_array_t * sobol_array,
int nb_parameters,
int vect_size )

```

This function initialise a Martinez Sobol indices structure

#### Parameters

in, out	<i>*sobol_array</i>	input: reference or pointer to an uninitialised sobol indices structure, output: initialised structure, with values and variances set to 0
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>vect_size</i>	size of the input vectors

#### 6.25.2.9 read\_sobol\_jansen()

```

void read_sobol_jansen (
    sobol_array_t * sobol_array,
    int vect_size,
    int nb_time_steps,
    int nb_parameters,
    FILE * f )

```

This function reads an array of sobol\_jansen structures on disc

#### Parameters

in	<i>*sobol_array</i>	sobol_array structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>f</i>	file descriptor

#### 6.25.2.10 read\_sobol\_martinez()

```

void read_sobol_martinez (
    sobol_array_t * sobol_array,
    int vect_size,
    int nb_time_steps,
    int nb_parameters,
    FILE * f )

```

This function reads an array of sobol\_martinez structures on disc

#### Parameters

in	<i>*sobol_array</i>	sobol_array structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors



**Parameters**

in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>f</i>	file descriptor

**6.25.2.11 save\_sobol\_jansen()**

```
void save_sobol_jansen (
    sobol_array_t * sobol_array,
    int vect_size,
    int nb_time_steps,
    int nb_parameters,
    FILE * f )
```

This function writes an array of sobol\_jansen structures on disc

**Parameters**

in	<i>*sobol_array</i>	sobol_array structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>f</i>	file descriptor

**6.25.2.12 save\_sobol\_martinez()**

```
void save_sobol_martinez (
    sobol_array_t * sobol_array,
    int vect_size,
    int nb_time_steps,
    int nb_parameters,
    FILE * f )
```

This function writes an array of sobol\_martinez structures on disc

**Parameters**

in	<i>*sobol_array</i>	sobol_array structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>f</i>	file descriptor

## 6.26 /home/tterraz/avido/melissa/Melissa/source/stats/sobol.h File Reference

This graph shows which files directly or indirectly include this file:

### Classes

- struct [sobol\\_jansen\\_s](#)
- struct [sobol\\_martinez\\_s](#)
- struct [sobol\\_array\\_s](#)

### Typedefs

- typedef struct [sobol\\_jansen\\_s](#) [sobol\\_jansen\\_t](#)
- typedef struct [sobol\\_martinez\\_s](#) [sobol\\_martinez\\_t](#)
- typedef struct [sobol\\_array\\_s](#) [sobol\\_array\\_t](#)

### Functions

- void [init\\_sobol\\_jansen](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters, int vect\_size)
- void [init\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters, int vect\_size)
- void [increment\\_sobol\\_jansen](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters, double \*\*in\_vect\_tab, int vect\_size)
- void [increment\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters, double \*\*in\_vect\_tab, int vect\_size)
- void [confidence\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters, int vect\_size)
- int [check\\_convergence\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*\*sobol\_array, double confidence\_value, int nb\_time\_steps, int nb\_parameters)
- void [save\\_sobol\\_jansen](#) ([sobol\\_array\\_t](#) \*sobol\_array, int vect\_size, int nb\_time\_steps, int nb\_parameters, FILE \*f)
- void [save\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int vect\_size, int nb\_time\_steps, int nb\_parameters, FILE \*f)
- void [read\\_sobol\\_jansen](#) ([sobol\\_array\\_t](#) \*sobol\_array, int vect\_size, int nb\_time\_steps, int nb\_parameters, FILE \*f)
- void [read\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int vect\_size, int nb\_time\_steps, int nb\_parameters, FILE \*f)
- void [free\\_sobol\\_jansen](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters)
- void [free\\_sobol\\_martinez](#) ([sobol\\_array\\_t](#) \*sobol\_array, int nb\_parameters)

### 6.26.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-15-01

## 6.26.2 Typedef Documentation

### 6.26.2.1 sobol\_array\_t

```
typedef struct sobol_array_s sobol_array_t
```

type corresponding to [sobol\\_array\\_s](#)

### 6.26.2.2 sobol\_jansen\_t

```
typedef struct sobol_jansen_s sobol_jansen_t
```

type corresponding to [sobol\\_martinez\\_s](#)

### 6.26.2.3 sobol\_martinez\_t

```
typedef struct sobol_martinez_s sobol_martinez_t
```

type corresponding to [sobol\\_martinez\\_s](#)

## 6.26.3 Function Documentation

### 6.26.3.1 check\_convergence\_sobol\_martinez()

```
int check_convergence_sobol_martinez (  
    sobol\_array\_t ** sobol_array,  
    double confidence_value,  
    int nb_time_steps,  
    int nb_parameters )
```

This function check if the Sobol indice convergence has been reached

#### Parameters

out	** <i>sobol_array</i>	Sobol indices
in	<i>confidence_value</i>	value to reach for the worst confidence interval
in	<i>nb_time_steps</i>	number of time steps of the study
in	<i>nb_parameters</i>	size of <i>sobol_array-&gt;sobol_martinez</i>

**Returns**

[out] int 0 if convergence is not reached 1 if convergence is reached

**6.26.3.2 confidence\_sobol\_martinez()**

```
void confidence_sobol_martinez (
    sobol_array_t * sobol_array,
    int nb_parameters,
    int vect_size )
```

This function computes the confidence interval for Martinez Sobol indices

**Parameters**

out	<i>*sobol_array</i>	Sobol indices
in	<i>nb_parameters</i>	size of sobol_array->sobol_martinez
in	<i>vect_size</i>	size of input vectors

**6.26.3.3 free\_sobol\_jansen()**

```
void free_sobol_jansen (
    sobol_array_t * sobol_array,
    int nb_parameters )
```

This function frees a Jansen Sobol array structure

**Parameters**

in	<i>*sobol_array</i>	reference or pointer to a sobol index structure to free
in	<i>nb_parameters</i>	number of parameters of the study

**6.26.3.4 free\_sobol\_martinez()**

```
void free_sobol_martinez (
    sobol_array_t * sobol_array,
    int nb_parameters )
```

This function frees a Martinez Sobol indices structure

**Parameters**

in	<i>*sobol_array</i>	reference or pointer to a sobol array structure to free
in	<i>nb_parameters</i>	number of parameters of the study

#### 6.26.3.5 increment\_sobol\_jansen()

```
void increment_sobol_jansen (
    sobol_array_t * sobol_array,
    int nb_parameters,
    double ** in_vect_tab,
    int vect_size )
```

This function computes Sobol indices using Jansen formula

##### Parameters

out	<i>*sobol_array</i>	computed sobol indices, using Jansen formula
in	<i>nb_parameters</i>	size of sobol_array->sobol_jansen
in	<i>**in_vect_tab</i>	array of input vectors
in	<i>vect_size</i>	size of input vectors

#### 6.26.3.6 increment\_sobol\_martinez()

```
void increment_sobol_martinez (
    sobol_array_t * sobol_array,
    int nb_parameters,
    double ** in_vect_tab,
    int vect_size )
```

This function computes Sobol indices using Martinez formula

##### Parameters

out	<i>*sobol_array</i>	computed sobol indices, using Martinez formula
in	<i>nb_parameters</i>	size of sobol_array->sobol_martinez
in	<i>**in_vect_tab</i>	array of input vectors
in	<i>vect_size</i>	size of input vectors

#### 6.26.3.7 init\_sobol\_jansen()

```
void init_sobol_jansen (
    sobol_array_t * sobol_array,
    int nb_parameters,
    int vect_size )
```

This function initialise a Jansen Sobol indices structure

**Parameters**

in, out	<i>*sobol_array</i>	input: reference or pointer to an uninitialised sobol indices structure, output: initialised structure, with values and variances set to 0
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>vect_size</i>	size of the input vectors

**6.26.3.8 init\_sobol\_martinez()**

```
void init_sobol_martinez (
    sobol_array_t * sobol_array,
    int nb_parameters,
    int vect_size )
```

This function initialise a Martinez Sobol indices structure

**Parameters**

in, out	<i>*sobol_array</i>	input: reference or pointer to an uninitialised sobol indices structure, output: initialised structure, with values and variances set to 0
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>vect_size</i>	size of the input vectors

**6.26.3.9 read\_sobol\_jansen()**

```
void read_sobol_jansen (
    sobol_array_t * sobol_array,
    int vect_size,
    int nb_time_steps,
    int nb_parameters,
    FILE * f )
```

This function reads an array of sobol\_jansen structures on disc

**Parameters**

in	<i>*sobol_array</i>	sobol_array structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>f</i>	file descriptor

## 6.26.3.10 read\_sobol\_martinez()

```
void read_sobol_martinez (
    sobol_array_t * sobol_array,
    int vect_size,
    int nb_time_steps,
    int nb_parameters,
    FILE * f )
```

This function reads an array of sobol\_martinez structures on disc

## Parameters

in	<i>*sobol_array</i>	sobol_array structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>f</i>	file descriptor

## 6.26.3.11 save\_sobol\_jansen()

```
void save_sobol_jansen (
    sobol_array_t * sobol_array,
    int vect_size,
    int nb_time_steps,
    int nb_parameters,
    FILE * f )
```

This function writes an array of sobol\_jansen structures on disc

## Parameters

in	<i>*sobol_array</i>	sobol_array structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>f</i>	file descriptor

## 6.26.3.12 save\_sobol\_martinez()

```
void save_sobol_martinez (
    sobol_array_t * sobol_array,
    int vect_size,
    int nb_time_steps,
    int nb_parameters,
    FILE * f )
```

This function writes an array of `sobol_martinez` structures on disc



## Parameters

in	<i>*sobel_array</i>	sobel_array structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>nb_parameters</i>	number of parameters of the study
in	<i>f</i>	file descriptor

## 6.27 /home/tterraz/avido/melissa/Melissa/source/stats/threshold.c File Reference

Threshold exceedance related functions.

```
#include <stdlib.h>
#include <stdio.h>
#include "threshold.h"
#include "melissa_utils.h"
Include dependency graph for threshold.c:
```

### Functions

- void [update\\_threshold\\_exceedance](#) (int threshold\_exceedance[], double threshold, double in\_vect[], const int vect\_size)
- void [save\\_threshold](#) (int \*\*threshold\_exceedance, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_threshold](#) (int \*\*threshold\_exceedance, int vect\_size, int nb\_time\_steps, FILE \*f)

#### 6.27.1 Detailed Description

Threshold exceedance related functions.

##### Author

Terraz Théophile

##### Date

2016-15-02

#### 6.27.2 Function Documentation

##### 6.27.2.1 read\_threshold()

```
void read_threshold (
    int ** threshold_exceedance,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of threshold exceedance vectors on disc

**Parameters**

in	<i>**threshold_exceedance</i>	threshold exceedance array of vectors to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.27.2.2 save\_threshold()**

```
void save_threshold (
    int ** threshold_exceedance,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function writes an array of threshold exceedance vectors on disc

**Parameters**

in	<i>**threshold_exceedance</i>	threshold exceedance array of vectors to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.27.2.3 update\_threshold\_exceedance()**

```
void update_threshold_exceedance (
    int threshold_exceedance[],
    double threshold,
    double in_vect[],
    const int vect_size )
```

This function updates the number of values exceeding a given threshold

**Parameters**

in, out	<i>threshold_exceedance[]</i>	number of threshold exceedance occurrences
in	<i>threshold</i>	
in	<i>in_vect[]</i>	input vector of double values
in	<i>vect_size</i>	size of the input vector

## 6.28 /home/tterraz/avido/melissa/Melissa/source/stats/threshold.h File Reference

This graph shows which files directly or indirectly include this file:

### Functions

- void [update\\_threshold\\_exceedance](#) (int threshold\_exceedance[], double threshold, double in\_vect[], const int vect\_size)
- void [save\\_threshold](#) (int \*\*threshold\_exceedance, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_threshold](#) (int \*\*threshold\_exceedance, int vect\_size, int nb\_time\_steps, FILE \*f)

### 6.28.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-15-01

### 6.28.2 Function Documentation

#### 6.28.2.1 read\_threshold()

```
void read_threshold (
    int ** threshold_exceedance,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of threshold exceedance vectors on disc

#### Parameters

in	<b><i>**threshold_exceedance</i></b>	threshold exceedance array of vectors to read, size nb_time_steps
in	<b><i>vect_size</i></b>	size of double vectors
in	<b><i>nb_time_steps</i></b>	number of time_steps of the study
in	<b><i>f</i></b>	file descriptor

#### 6.28.2.2 save\_threshold()

```
void save_threshold (
```

```

    int ** threshold_exceedance,
    int vect_size,
    int nb_time_steps,
    FILE * f )

```

This function writes an array of threshold exceedance vectors on disc

#### Parameters

in	<b><i>**threshold_exceedance</i></b>	threshold exceedance array of vectors to save, size nb_time_steps
in	<b><i>vect_size</i></b>	size of double vectors
in	<b><i>nb_time_steps</i></b>	number of time_steps of the study
in	<b><i>f</i></b>	file descriptor

#### 6.28.2.3 update\_threshold\_exceedance()

```

void update_threshold_exceedance (
    int threshold_exceedance[],
    double threshold,
    double in_vect[],
    const int vect_size )

```

This function updates the number of values exceeding a given threshold

#### Parameters

in, out	<b><i>threshold_exceedance[]</i></b>	number of threshold exceedance occurrences
in	<b><i>threshold</i></b>	
in	<b><i>in_vect[]</i></b>	input vector of double values
in	<b><i>vect_size</i></b>	size of the input vector

## 6.29 /home/tterraz/avido/melissa/Melissa/source/stats/variance.c File Reference

Variance related functions.

```

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "mean.h"
#include "variance.h"
#include "melissa_utils.h"

```

Include dependency graph for variance.c:

#### Functions

- void [init\\_variance](#) ([variance\\_t](#) \*variance, const int vect\_size)

- void [increment\\_mean\\_and\\_variance](#) ([variance\\_t](#) \*partial\_variance, double in\_vect[], const int vect\_size)
- void [increment\\_variance](#) ([variance\\_t](#) \*partial\_variance, double in\_vect[], const int vect\_size)
- void [update\\_variance](#) ([variance\\_t](#) \*variance1, [variance\\_t](#) \*variance2, [variance\\_t](#) \*updated\_variance, const int vect\_size)
- void [save\\_variance](#) ([variance\\_t](#) \*vars, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_variance](#) ([variance\\_t](#) \*vars, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [free\\_variance](#) ([variance\\_t](#) \*variance)

## 6.29.1 Detailed Description

Variance related functions.

### Author

Terraz Théophile

### Date

2016-15-02

## 6.29.2 Function Documentation

### 6.29.2.1 [free\\_variance\(\)](#)

```
void free_variance (  
    variance\_t * variance )
```

This function frees a variance structure.

#### Parameters

in	* <i>variance</i>	the variance structure to free
----	-------------------	--------------------------------

### 6.29.2.2 [increment\\_mean\\_and\\_variance\(\)](#)

```
void increment_mean_and_variance (  
    variance\_t * partial_variance,  
    double in_vect[],  
    const int vect_size )
```

This function updates the incremental mean and variance.

**Parameters**

in, out	<i>*partial_variance</i>	input: previously computed partial variance, output: updated partial variance
in	<i>in_vect[]</i>	input vector of double values
in	<i>vect_size</i>	size of the input vectors

**6.29.2.3 increment\_variance()**

```
void increment_variance (
    variance_t * partial_variance,
    double in_vect[],
    const int vect_size )
```

This function updates the incremental variance.

**Parameters**

in, out	<i>*partial_variance</i>	input: previously computed partial variance, output: updated partial variance
in	<i>in_vect[]</i>	input vector of double values
in	<i>vect_size</i>	size of the input vectors

**6.29.2.4 init\_variance()**

```
void init_variance (
    variance_t * variance,
    const int vect_size )
```

This function initializes a variance structure.

**Parameters**

in, out	<i>*variance</i>	the variance structure to initialize
in	<i>vect_size</i>	size of the variance vector

**6.29.2.5 read\_variance()**

```
void read_variance (
    variance_t * vars,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of variances structures on disc

**Parameters**

in	<i>*vars</i>	variance structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.29.2.6 save\_variance()**

```
void save_variance (
    variance_t * vars,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function writes an array of variances structures on disc

**Parameters**

in	<i>*vars</i>	variance structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.29.2.7 update\_variance()**

```
void update_variance (
    variance_t * variance1,
    variance_t * variance2,
    variance_t * updated_variance,
    const int vect_size )
```

This function aggregates two partial variances.

**Parameters**

in	<i>*variance1</i>	first input partial variances
in	<i>*variance2</i>	second input partial variances
out	<i>*updated_variance</i>	the updated variances
in	<i>vect_size</i>	size of the input and output vectors

## 6.30 /home/tterraz/avido/melissa/Melissa/source/stats/variance.h File Reference

This graph shows which files directly or indirectly include this file:

### Classes

- struct [variance\\_s](#)

### Typedefs

- typedef struct [variance\\_s](#) [variance\\_t](#)

### Functions

- void [init\\_variance](#) ([variance\\_t](#) \*variance, const int vect\_size)
- void [increment\\_mean\\_and\\_variance](#) ([variance\\_t](#) \*partial\_variance, double in\_vect[], const int vect\_size)
- void [increment\\_variance](#) ([variance\\_t](#) \*partial\_variance, double in\_vect[], const int vect\_size)
- void [update\\_variance](#) ([variance\\_t](#) \*variance1, [variance\\_t](#) \*variance2, [variance\\_t](#) \*updated\_variance, const int vect\_size)
- void [save\\_variance](#) ([variance\\_t](#) \*vars, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [read\\_variance](#) ([variance\\_t](#) \*vars, int vect\_size, int nb\_time\_steps, FILE \*f)
- void [free\\_variance](#) ([variance\\_t](#) \*variance)

### 6.30.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2017-15-01

### 6.30.2 Typedef Documentation

#### 6.30.2.1 [variance\\_t](#)

```
typedef struct variance\_s variance\_t
```

type corresponding to [variance\\_s](#)

### 6.30.3 Function Documentation

#### 6.30.3.1 [free\\_variance\(\)](#)

```
void free_variance (  
    variance\_t * variance )
```

This function frees a variance structure.



## Parameters

in	<i>*variance</i>	the variance structure to free
----	------------------	--------------------------------

## 6.30.3.2 increment\_mean\_and\_variance()

```
void increment_mean_and_variance (
    variance_t * partial_variance,
    double in_vect[],
    const int vect_size )
```

This function updates the incremental mean and variance.

## Parameters

in, out	<i>*partial_variance</i>	input: previously computed partial variance, output: updated partial variance
in	<i>in_vect[]</i>	input vector of double values
in	<i>vect_size</i>	size of the input vectors

## 6.30.3.3 increment\_variance()

```
void increment_variance (
    variance_t * partial_variance,
    double in_vect[],
    const int vect_size )
```

This function updates the incremental variance.

## Parameters

in, out	<i>*partial_variance</i>	input: previously computed partial variance, output: updated partial variance
in	<i>in_vect[]</i>	input vector of double values
in	<i>vect_size</i>	size of the input vectors

## 6.30.3.4 init\_variance()

```
void init_variance (
    variance_t * variance,
    const int vect_size )
```

This function initializes a variance structure.

**Parameters**

in, out	<i>*variance</i>	the variance structure to initialize
in	<i>vect_size</i>	size of the variance vector

**6.30.3.5 read\_variance()**

```
void read_variance (
    variance_t * vars,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function reads an array of variances structures on disc

**Parameters**

in	<i>*vars</i>	variance structures to read, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.30.3.6 save\_variance()**

```
void save_variance (
    variance_t * vars,
    int vect_size,
    int nb_time_steps,
    FILE * f )
```

This function writes an array of variances structures on disc

**Parameters**

in	<i>*vars</i>	variance structures to save, size nb_time_steps
in	<i>vect_size</i>	size of double vectors
in	<i>nb_time_steps</i>	number of time_steps of the study
in	<i>f</i>	file descriptor

**6.30.3.7 update\_variance()**

```
void update_variance (
    variance_t * variance1,
```

```

    variance_t * variance2,
    variance_t * updated_variance,
    const int vect_size )

```

This function aggregates two partial variances.

#### Parameters

in	<i>*variance1</i>	first input partial variances
in	<i>*variance2</i>	second input partial variances
out	<i>*updated_variance</i>	the updated variances
in	<i>vect_size</i>	size of the input and output vectors

## 6.31 /home/tterraz/avido/melissa/Melissa/source/utls/melissa\_utils.c File Reference

Functions used in Melissa.

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdint.h>
#include <time.h>
#include <sys/timeb.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <ifaddrs.h>
#include <zmq.h>
#include "melissa_utils.h"
Include dependency graph for melissa_utils.c:

```

#### Functions

- void [melissa\\_logo](#) ()
- void \* [melissa\\_malloc](#) (size\_t size)
- void \* [melissa\\_calloc](#) (size\_t num, size\_t size)
- void [melissa\\_free](#) (void \*ptr)
- void [melissa\\_bind](#) (void \*socket, char \*port\_name)
- void [melissa\\_connect](#) (void \*socket, char \*port\_name)
- double [melissa\\_get\\_time](#) ()
- void [melissa\\_get\\_node\\_name](#) (char \*node\_name)
- void [set\\_bit](#) (int32\_t \*vect, int pos)
- void [clear\\_bit](#) (int32\_t \*vect, int pos)
- int [test\\_bit](#) (int32\_t \*vect, int pos)

### 6.31.1 Detailed Description

Functions used in Melissa.

#### Author

Terraz Théophile

#### Date

2017-15-01

## 6.32 /home/tterraz/avido/melissa/Melissa/source/utils/melissa\_utils.h File Reference

```
#include <stdio.h>
```

Include dependency graph for melissa\_utils.h: This graph shows which files directly or indirectly include this file:

### Macros

- `#define MPI_MAX_PROCESSOR_NAME` 256
- `#define MAX_FIELD_NAME` 128

### Functions

- void `melissa_logo` ()
- void \* `melissa_malloc` (size\_t size)
- void \* `melissa_calloc` (size\_t num, size\_t size)
- void `melissa_free` (void \*ptr)
- void `melissa_bind` (void \*socket, char \*port\_name)
- void `melissa_connect` (void \*socket, char \*port\_name)
- double `melissa_get_time` ()
- void `melissa_get_node_name` (char \*node\_name)
- void `set_bit` (int32\_t \*vect, int pos)
- void `clear_bit` (int32\_t \*vect, int pos)
- int `test_bit` (int32\_t \*vect, int pos)

### 6.32.1 Detailed Description

#### Author

Terraz Théophile

#### Date

2016-15-02

## 6.32.2 Macro Definition Documentation

### 6.32.2.1 MAX\_FIELD\_NAME

```
#define MAX_FIELD_NAME 128
```

Define name size if not defined

### 6.32.2.2 MPI\_MAX\_PROCESSOR\_NAME

```
#define MPI_MAX_PROCESSOR_NAME 256
```

Define the macro if mpi.h not present



# Index

/home/tterraz/avido/melissa/Melissa/source/api/melissa↔ c, 84  
\_api.c, 49 /home/tterraz/avido/melissa/Melissa/source/stats/sobol.↔  
/home/tterraz/avido/melissa/Melissa/source/api/melissa↔ h, 90  
\_api.h, 49 /home/tterraz/avido/melissa/Melissa/source/stats/threshold.↔  
/home/tterraz/avido/melissa/Melissa/source/api/melissa↔ c, 97  
\_api\_no\_mpi.h, 49 /home/tterraz/avido/melissa/Melissa/source/stats/threshold.↔  
/home/tterraz/avido/melissa/Melissa/source/server/compute↔ h, 99  
\_stats.c, 50 /home/tterraz/avido/melissa/Melissa/source/stats/variance.↔  
/home/tterraz/avido/melissa/Melissa/source/server/compute↔ c, 100  
\_stats.h, 50 /home/tterraz/avido/melissa/Melissa/source/stats/variance.↔  
/home/tterraz/avido/melissa/Melissa/source/server/fault↔ h, 104  
\_tolerance.c, 51 /home/tterraz/avido/melissa/Melissa/source/utils/melissa↔  
/home/tterraz/avido/melissa/Melissa/source/server/fault↔ \_utils.c, 107  
\_tolerance.h, 51 /home/tterraz/avido/melissa/Melissa/source/utils/melissa↔  
/home/tterraz/avido/melissa/Melissa/source/server/melissa↔ \_utils.h, 108  
\_data.c, 52  
/home/tterraz/avido/melissa/Melissa/source/server/melissa↔ API, 21  
\_data.h, 53 melissa\_finalize, 21  
/home/tterraz/avido/melissa/Melissa/source/server/melissa↔ melissa\_init, 21  
\_fields.c, 55 melissa\_init\_no\_mpi, 22  
/home/tterraz/avido/melissa/Melissa/source/server/melissa↔ melissa\_send, 22  
\_fields.h, 56 melissa\_send\_no\_mpi, 22  
/home/tterraz/avido/melissa/Melissa/source/server/melissa↔ alpha  
\_io.c, 57 quantile\_s, 38  
/home/tterraz/avido/melissa/Melissa/source/server/melissa↔ buff\_size  
\_io.h, 58 pull\_data\_s, 36  
/home/tterraz/avido/melissa/Melissa/source/server/melissa↔ zmq\_data\_s, 43  
\_options.c, 58  
/home/tterraz/avido/melissa/Melissa/source/server/melissa↔ buffer  
\_options.h, 59 zmq\_data\_s, 44  
/home/tterraz/avido/melissa/Melissa/source/server/server.↔ buffer\_sobol  
h, 60 zmq\_data\_s, 44  
/home/tterraz/avido/melissa/Melissa/source/stats/covariance↔ check\_convergence\_sobol\_martinez  
c, 62 sobol.c, 85  
/home/tterraz/avido/melissa/Melissa/source/stats/covariance.↔ sobol.h, 91  
h, 65  
/home/tterraz/avido/melissa/Melissa/source/stats/mean.↔ clear\_bit  
c, 68 misc functions, 7  
/home/tterraz/avido/melissa/Melissa/source/stats/mean.↔ client\_comm\_size  
h, 70 comm\_data\_s, 25  
/home/tterraz/avido/melissa/Melissa/source/stats/min↔ comm\_data\_s, 25  
\_max.c, 74 client\_comm\_size, 25  
/home/tterraz/avido/melissa/Melissa/source/stats/min↔ comm\_size, 25  
\_max.h, 76 rank, 25  
/home/tterraz/avido/melissa/Melissa/source/stats/quantile.↔ rcounts, 26  
c, 79 rdispls, 26  
/home/tterraz/avido/melissa/Melissa/source/stats/quantile.↔ comm\_data\_t  
h, 81 melissa\_data.h, 54  
/home/tterraz/avido/melissa/Melissa/source/stats/sobol.↔ comm\_size  
comm\_data\_s, 25

- comm\_sobol
  - zmq\_data\_s, [44](#)
- compute\_stats
  - internal API, [11](#)
- confidence\_interval
  - sobol\_martinez\_s, [41](#)
- confidence\_sobol\_martinez
  - sobol.c, [85](#)
  - sobol.h, [92](#)
- connexion\_requester
  - zmq\_data\_s, [44](#)
- context
  - zmq\_data\_s, [44](#)
- coupling
  - zmq\_data\_s, [44](#)
- covariance
  - covariance\_s, [26](#)
- covariance.c
  - free\_covariance, [62](#)
  - increment\_covariance, [63](#)
  - init\_covariance, [63](#)
  - read\_covariance, [63](#)
  - save\_covariance, [64](#)
  - update\_covariance, [64](#)
- covariance.h
  - covariance\_t, [65](#)
  - free\_covariance, [65](#)
  - increment\_covariance, [66](#)
  - init\_covariance, [66](#)
  - read\_covariance, [66](#)
  - save\_covariance, [67](#)
  - update\_covariance, [67](#)
- covariance\_s, [26](#)
  - covariance, [26](#)
  - increment, [27](#)
  - mean1, [27](#)
  - mean2, [27](#)
- covariance\_t
  - covariance.h, [65](#)
- data\_pusher
  - zmq\_data\_s, [44](#)
- finalize\_stats
  - internal API, [11](#)
- first\_order\_covariance
  - sobol\_martinez\_s, [41](#)
- first\_order\_values
  - sobol\_jansen\_s, [40](#)
  - sobol\_martinez\_s, [41](#)
- free\_covariance
  - covariance.c, [62](#)
  - covariance.h, [65](#)
- free\_mean
  - mean.c, [68](#)
  - mean.h, [71](#)
- free\_min\_max
  - min\_max.c, [74](#)
  - min\_max.h, [77](#)
- free\_quantile
  - quantile.c, [79](#)
  - quantile.h, [82](#)
- free\_sobol
  - melissa\_data\_s, [28](#)
- free\_sobol\_jansen
  - sobol.c, [86](#)
  - sobol.h, [92](#)
- free\_sobol\_martinez
  - sobol.c, [86](#)
  - sobol.h, [92](#)
- free\_variance
  - variance.c, [101](#)
  - variance.h, [104](#)
- Get options from command line, [19](#)
  - melissa\_check\_options, [19](#)
  - melissa\_get\_options, [19](#)
  - melissa\_print\_options, [19](#)
  - melissa\_read\_options, [20](#)
  - melissa\_write\_options, [20](#)
- global\_confidence\_sobol\_martinez
  - server.h, [61](#)
- global\_vect\_size
  - melissa\_options\_s, [32](#)
- increment
  - covariance\_s, [27](#)
  - mean\_s, [27](#)
  - quantile\_s, [38](#)
- increment\_covariance
  - covariance.c, [63](#)
  - covariance.h, [66](#)
- increment\_mean
  - mean.c, [69](#)
  - mean.h, [72](#)
- increment\_mean\_and\_variance
  - variance.c, [101](#)
  - variance.h, [105](#)
- increment\_quantile
  - quantile.c, [80](#)
  - quantile.h, [82](#)
- increment\_sobol
  - melissa\_data\_s, [29](#)
- increment\_sobol\_jansen
  - sobol.c, [86](#)
  - sobol.h, [93](#)
- increment\_sobol\_martinez
  - sobol.c, [87](#)
  - sobol.h, [93](#)
- increment\_variance
  - variance.c, [102](#)
  - variance.h, [105](#)
- init\_covariance
  - covariance.c, [63](#)
  - covariance.h, [66](#)
- init\_mean
  - mean.c, [69](#)
  - mean.h, [72](#)



- init\_min\_max
  - min\_max.c, [74](#)
  - min\_max.h, [77](#)
- init\_quantile
  - quantile.c, [80](#)
  - quantile.h, [83](#)
- init\_requester
  - zmq\_data\_s, [44](#)
- init\_sobol
  - melissa\_data\_s, [29](#)
- init\_sobol\_jansen
  - sobol.c, [87](#)
  - sobol.h, [93](#)
- init\_sobol\_martinez
  - sobol.c, [87](#)
  - sobol.h, [94](#)
- init\_variance
  - variance.c, [102](#)
  - variance.h, [105](#)
- input, output and checkpoint functions, [14](#)
  - read\_client\_data, [14](#)
  - read\_ensight, [14](#)
  - read\_saved\_stats, [15](#)
  - read\_simu\_states, [15](#)
  - save\_simu\_states, [16](#)
  - save\_stats, [16](#)
  - write\_client\_data, [16](#)
  - write\_stats\_bin, [17](#)
  - write\_stats\_ensight, [17](#)
  - write\_stats\_txt, [17](#)
- internal API, [11](#)
  - compute\_stats, [11](#)
  - finalize\_stats, [11](#)
- is\_init
  - min\_max\_s, [35](#)
- is\_valid
  - melissa\_data\_s, [29](#)
- iteration
  - sobol\_array\_s, [39](#)
- launcher\_name
  - melissa\_options\_s, [32](#)
- local\_nb\_messages
  - pull\_data\_s, [36](#)
  - zmq\_data\_s, [45](#)
- local\_vect\_sizes
  - zmq\_data\_s, [45](#)
- MAX\_FIELD\_NAME
  - melissa\_utils.h, [109](#)
- MPI\_MAX\_PROCESSOR\_NAME
  - melissa\_utils.h, [109](#)
- max
  - min\_max\_s, [35](#)
- mean
  - mean\_s, [28](#)
- mean.c
  - free\_mean, [68](#)
  - increment\_mean, [69](#)
- init\_mean, [69](#)
- read\_mean, [69](#)
- save\_mean, [70](#)
- update\_mean, [70](#)
- mean.h
  - free\_mean, [71](#)
  - increment\_mean, [72](#)
  - init\_mean, [72](#)
  - mean\_t, [71](#)
  - read\_mean, [72](#)
  - save\_mean, [73](#)
  - update\_mean, [73](#)
- mean1
  - covariance\_s, [27](#)
- mean2
  - covariance\_s, [27](#)
- mean\_op
  - melissa\_options\_s, [32](#)
- mean\_s, [27](#)
  - increment, [27](#)
  - mean, [28](#)
- mean\_structure
  - variance\_s, [42](#)
- mean\_t
  - mean.h, [71](#)
- means
  - melissa\_data\_s, [29](#)
- Melissa data, [13](#)
  - melissa\_free\_data, [13](#)
  - melissa\_init\_data, [13](#)
- Melissa fields, [12](#)
- melissa\_bind
  - misc functions, [8](#)
- melissa\_calloc
  - misc functions, [8](#)
- melissa\_check\_data
  - melissa\_data.c, [53](#)
  - melissa\_data.h, [55](#)
- melissa\_check\_options
  - Get options from command line, [19](#)
- melissa\_connect
  - misc functions, [8](#)
- melissa\_data.c
  - melissa\_check\_data, [53](#)
  - mem\_conso, [53](#)
- melissa\_data.h
  - comm\_data\_t, [54](#)
  - melissa\_check\_data, [55](#)
  - melissa\_data\_t, [54](#)
  - mem\_conso, [55](#)
- melissa\_data\_s, [28](#)
  - free\_sobol, [28](#)
  - increment\_sobol, [29](#)
  - init\_sobol, [29](#)
  - is\_valid, [29](#)
  - means, [29](#)
  - min\_max, [29](#)
  - nb\_simu, [29](#)

- options, 29
- quantiles, 29
- read\_sobol, 30
- save\_sobol, 30
- sobol\_indices, 30
- step\_simu, 30
- thresholds, 30
- variances, 30
- vect\_size, 30
- melissa\_data\_t
  - melissa\_data.h, 54
- melissa\_field\_s, 31
  - name, 31
  - stats\_data, 31
- melissa\_field\_t
  - melissa\_fields.h, 57
- melissa\_fields.h
  - melissa\_field\_t, 57
- melissa\_finalize
  - API, 21
- melissa\_free
  - misc functions, 9
- melissa\_free\_data
  - Melissa data, 13
- melissa\_get\_node\_name
  - misc functions, 9
- melissa\_get\_options
  - Get options from command line, 19
- melissa\_get\_time
  - misc functions, 9
- melissa\_init
  - API, 21
- melissa\_init\_data
  - Melissa data, 13
- melissa\_init\_no\_mpi
  - API, 22
- melissa\_logo
  - misc functions, 9
- melissa\_malloc
  - misc functions, 9
- melissa\_options.h
  - melissa\_options\_t, 60
- melissa\_options\_s, 32
  - global\_vect\_size, 32
  - launcher\_name, 32
  - mean\_op, 32
  - min\_and\_max\_op, 33
  - nb\_fields, 33
  - nb\_parameters, 33
  - nb\_simu, 33
  - nb\_time\_steps, 33
  - quantile\_op, 33
  - restart, 33
  - restart\_dir, 33
  - sampling\_size, 34
  - sobol\_op, 34
  - sobol\_order, 34
  - threshold, 34
  - threshold\_op, 34
  - variance\_op, 34
- melissa\_options\_t
  - melissa\_options.h, 60
- melissa\_print\_options
  - Get options from command line, 19
- melissa\_read\_options
  - Get options from command line, 20
- melissa\_send
  - API, 22
- melissa\_send\_no\_mpi
  - API, 22
- melissa\_simulation\_s, 35
- melissa\_utils.h
  - MAX\_FIELD\_NAME, 109
  - MPI\_MAX\_PROCESSOR\_NAME, 109
- melissa\_write\_options
  - Get options from command line, 20
- mem\_conso
  - melissa\_data.c, 53
  - melissa\_data.h, 55
- message\_sizes
  - pull\_data\_s, 36
  - zmq\_data\_s, 45
- min
  - min\_max\_s, 35
- min\_and\_max
  - min\_max.c, 75
  - min\_max.h, 77
- min\_and\_max\_op
  - melissa\_options\_s, 33
- min\_max
  - melissa\_data\_s, 29
- min\_max.c
  - free\_min\_max, 74
  - init\_min\_max, 74
  - min\_and\_max, 75
  - read\_min\_max, 75
  - save\_min\_max, 76
- min\_max.h
  - free\_min\_max, 77
  - init\_min\_max, 77
  - min\_and\_max, 77
  - min\_max\_t, 77
  - read\_min\_max, 78
  - save\_min\_max, 78
- min\_max\_s, 35
  - is\_init, 35
  - max, 35
  - min, 35
- min\_max\_t
  - min\_max.h, 77
- misc functions, 7
  - clear\_bit, 7
  - melissa\_bind, 8
  - melissa\_calloc, 8
  - melissa\_connect, 8
  - melissa\_free, 9

- melissa\_get\_node\_name, 9
- melissa\_get\_time, 9
- melissa\_logo, 9
- melissa\_malloc, 9
- set\_bit, 10
- test\_bit, 10
- name
  - melissa\_field\_s, 31
- nb\_fields
  - melissa\_options\_s, 33
- nb\_parameters
  - melissa\_options\_s, 33
  - zmq\_data\_s, 45
- nb\_proc\_server
  - zmq\_data\_s, 45
- nb\_simu
  - melissa\_data\_s, 29
  - melissa\_options\_s, 33
- nb\_time\_steps
  - melissa\_options\_s, 33
- options
  - melissa\_data\_s, 29
- pull\_data\_s, 36
  - buff\_size, 36
  - local\_nb\_messages, 36
  - message\_sizes, 36
  - pull\_rank, 37
  - push\_rank, 37
  - total\_nb\_messages, 37
- pull\_data\_t
  - server.h, 61
- pull\_rank
  - pull\_data\_s, 37
  - zmq\_data\_s, 45
- push\_rank
  - pull\_data\_s, 37
  - zmq\_data\_s, 45
- quantile
  - quantile\_s, 38
- quantile.c
  - free\_quantile, 79
  - increment\_quantile, 80
  - init\_quantile, 80
  - read\_quantile, 80
  - save\_quantile, 81
- quantile.h
  - free\_quantile, 82
  - increment\_quantile, 82
  - init\_quantile, 83
  - quantile\_t, 82
  - read\_quantile, 83
  - save\_quantile, 83
- quantile\_op
  - melissa\_options\_s, 33
- quantile\_s, 37
  - alpha, 38
  - increment, 38
  - quantile, 38
- quantile\_t
  - quantile.h, 82
- quantiles
  - melissa\_data\_s, 29
- rank
  - comm\_data\_s, 25
- rcounts
  - comm\_data\_s, 26
- rdispls
  - comm\_data\_s, 26
- read\_client\_data
  - input, output and checkpoint functions, 14
- read\_covariance
  - covariance.c, 63
  - covariance.h, 66
- read\_ensight
  - input, output and checkpoint functions, 14
- read\_mean
  - mean.c, 69
  - mean.h, 72
- read\_min\_max
  - min\_max.c, 75
  - min\_max.h, 78
- read\_quantile
  - quantile.c, 80
  - quantile.h, 83
- read\_saved\_stats
  - input, output and checkpoint functions, 15
- read\_simu\_states
  - input, output and checkpoint functions, 15
- read\_sobol
  - melissa\_data\_s, 30
- read\_sobol\_jansen
  - sobol.c, 88
  - sobol.h, 94
- read\_sobol\_martinez
  - sobol.c, 88
  - sobol.h, 94
- read\_threshold
  - threshold.c, 97
  - threshold.h, 99
- read\_variance
  - variance.c, 102
  - variance.h, 106
- restart
  - melissa\_options\_s, 33
- restart\_dir
  - melissa\_options\_s, 33
- rinit\_tab
  - zmq\_data\_s, 45
- sampling\_size
  - melissa\_options\_s, 34
- save\_covariance
  - covariance.c, 64

- covariance.h, 67
- save\_mean
  - mean.c, 70
  - mean.h, 73
- save\_min\_max
  - min\_max.c, 76
  - min\_max.h, 78
- save\_quantile
  - quantile.c, 81
  - quantile.h, 83
- save\_simu\_states
  - input, output and checkpoint functions, 16
- save\_sobol
  - melissa\_data\_s, 30
- save\_sobol\_jansen
  - sobol.c, 89
  - sobol.h, 95
- save\_sobol\_martinez
  - sobol.c, 89
  - sobol.h, 95
- save\_stats
  - input, output and checkpoint functions, 16
- save\_threshold
  - threshold.c, 98
  - threshold.h, 99
- save\_variance
  - variance.c, 103
  - variance.h, 106
- sdispls
  - zmq\_data\_s, 46
- send\_buff\_size
  - zmq\_data\_s, 46
- send\_counts
  - zmq\_data\_s, 46
- server.h
  - global\_confidence\_sobol\_martinez, 61
  - pull\_data\_t, 61
- server\_vect\_size
  - zmq\_data\_s, 46
- set\_bit
  - misc functions, 10
- sinit\_tab
  - zmq\_data\_s, 46
- sobol
  - zmq\_data\_s, 46
- sobol.c
  - check\_convergence\_sobol\_martinez, 85
  - confidence\_sobol\_martinez, 85
  - free\_sobol\_jansen, 86
  - free\_sobol\_martinez, 86
  - increment\_sobol\_jansen, 86
  - increment\_sobol\_martinez, 87
  - init\_sobol\_jansen, 87
  - init\_sobol\_martinez, 87
  - read\_sobol\_jansen, 88
  - read\_sobol\_martinez, 88
  - save\_sobol\_jansen, 89
  - save\_sobol\_martinez, 89
- sobol.h
  - check\_convergence\_sobol\_martinez, 91
  - confidence\_sobol\_martinez, 92
  - free\_sobol\_jansen, 92
  - free\_sobol\_martinez, 92
  - increment\_sobol\_jansen, 93
  - increment\_sobol\_martinez, 93
  - init\_sobol\_jansen, 93
  - init\_sobol\_martinez, 94
  - read\_sobol\_jansen, 94
  - read\_sobol\_martinez, 94
  - save\_sobol\_jansen, 95
  - save\_sobol\_martinez, 95
  - sobol\_array\_t, 91
  - sobol\_jansen\_t, 91
  - sobol\_martinez\_t, 91
- sobol\_array\_s, 38
  - iteration, 39
  - sobol\_jansen, 39
  - sobol\_martinez, 39
  - variance\_a, 39
  - variance\_b, 39
- sobol\_array\_t
  - sobol.h, 91
- sobol\_indices
  - melissa\_data\_s, 30
- sobol\_jansen
  - sobol\_array\_s, 39
- sobol\_jansen\_s, 39
  - first\_order\_values, 40
  - summ\_a, 40
  - summ\_b, 40
  - total\_order\_values, 40
- sobol\_jansen\_t
  - sobol.h, 91
- sobol\_martinez
  - sobol\_array\_s, 39
- sobol\_martinez\_s, 41
  - confidence\_interval, 41
  - first\_order\_covariance, 41
  - first\_order\_values, 41
  - total\_order\_covariance, 41
  - total\_order\_values, 41
  - variance\_k, 42
- sobol\_martinez\_t
  - sobol.h, 91
- sobol\_op
  - melissa\_options\_s, 34
- sobol\_order
  - melissa\_options\_s, 34
- sobol\_rank
  - zmq\_data\_s, 46
- sobol\_requester
  - zmq\_data\_s, 46
- stats\_data
  - melissa\_field\_s, 31
- step\_simu
  - melissa\_data\_s, 30

- summ\_a
  - sobol\_jansen\_s, 40
- summ\_b
  - sobol\_jansen\_s, 40
- test\_bit
  - misc functions, 10
- threshold
  - melissa\_options\_s, 34
- threshold.c
  - read\_threshold, 97
  - save\_threshold, 98
  - update\_threshold\_exceedance, 98
- threshold.h
  - read\_threshold, 99
  - save\_threshold, 99
  - update\_threshold\_exceedance, 100
- threshold\_op
  - melissa\_options\_s, 34
- thresholds
  - melissa\_data\_s, 30
- total\_nb\_messages
  - pull\_data\_s, 37
  - zmq\_data\_s, 47
- total\_order\_covariance
  - sobol\_martinez\_s, 41
- total\_order\_values
  - sobol\_jansen\_s, 40
  - sobol\_martinez\_s, 41
- update\_covariance
  - covariance.c, 64
  - covariance.h, 67
- update\_mean
  - mean.c, 70
  - mean.h, 73
- update\_threshold\_exceedance
  - threshold.c, 98
  - threshold.h, 100
- update\_variance
  - variance.c, 103
  - variance.h, 106
- variance
  - variance\_s, 42
- variance.c
  - free\_variance, 101
  - increment\_mean\_and\_variance, 101
  - increment\_variance, 102
  - init\_variance, 102
  - read\_variance, 102
  - save\_variance, 103
  - update\_variance, 103
- variance.h
  - free\_variance, 104
  - increment\_mean\_and\_variance, 105
  - increment\_variance, 105
  - init\_variance, 105
  - read\_variance, 106
  - save\_variance, 106
  - update\_variance, 106
  - variance\_t, 104
- variance\_a
  - sobol\_array\_s, 39
- variance\_b
  - sobol\_array\_s, 39
- variance\_k
  - sobol\_martinez\_s, 42
- variance\_op
  - melissa\_options\_s, 34
- variance\_s, 42
  - mean\_structure, 42
  - variance, 42
- variance\_t
  - variance.h, 104
- variances
  - melissa\_data\_s, 30
- vect\_size
  - melissa\_data\_s, 30
- write\_client\_data
  - input, output and checkpoint functions, 16
- write\_stats\_bin
  - input, output and checkpoint functions, 17
- write\_stats\_ensight
  - input, output and checkpoint functions, 17
- write\_stats\_txt
  - input, output and checkpoint functions, 17
- zmq\_data\_s, 43
  - buff\_size, 43
  - buffer, 44
  - buffer\_sobol, 44
  - comm\_sobol, 44
  - connexion\_requester, 44
  - context, 44
  - coupling, 44
  - data\_pusher, 44
  - init\_requester, 44
  - local\_nb\_messages, 45
  - local\_vect\_sizes, 45
  - message\_sizes, 45
  - nb\_parameters, 45
  - nb\_proc\_server, 45
  - pull\_rank, 45
  - push\_rank, 45
  - rinit\_tab, 45
  - sdispls, 46
  - send\_buff\_size, 46
  - send\_counts, 46
  - server\_vect\_size, 46
  - sinit\_tab, 46
  - sobol, 46
  - sobol\_rank, 46
  - sobol\_requester, 46
  - total\_nb\_messages, 47