

SpecSwap-RMC

1.0a0

Generated by Doxygen 1.7.2

Sat Apr 28 2012 20:33:15

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	BasisContainer Struct Reference	5
3.1.1	Detailed Description	6
3.1.2	Member Function Documentation	6
3.1.2.1	compareIndex	6
3.1.2.2	compareWeight	6
3.1.3	Member Data Documentation	6
3.1.3.1	index	6
3.1.3.2	name	6
3.1.3.3	weight	6
3.2	Convolute Struct Reference	7
3.2.1	Detailed Description	7
3.2.2	Member Data Documentation	7
3.2.2.1	no1	7
3.2.2.2	no2	7
3.2.2.3	no3	7
3.2.2.4	no4	7
3.3	CurveData Class Reference	8
3.3.1	Detailed Description	10
3.3.2	Constructor & Destructor Documentation	10
3.3.2.1	CurveData	10
3.3.2.2	CurveData	10
3.3.3	Member Function Documentation	10
3.3.3.1	accept	10
3.3.3.2	calculate_chi2	11
3.3.3.3	get_chi2	11
3.3.3.4	get_chi2_new	11
3.3.3.5	init	11
3.3.3.6	notify	12
3.3.3.7	print	12
3.3.3.8	print_to_file	12
3.3.3.9	read_reference	12
3.3.3.10	weighted_analysis	12

3.3.4	Friends And Related Function Documentation	13
3.3.4.1	Test_CurveData	13
3.3.5	Member Data Documentation	13
3.3.5.1	area_renorm_	13
3.3.5.2	chi2_	13
3.3.5.3	chi2_new_	13
3.3.5.4	curve_	13
3.3.5.5	curve_name_	13
3.3.5.6	curve_new_	14
3.3.5.7	curve_reference_	14
3.3.5.8	nsample_	14
3.3.5.9	one_over_sigma2_	14
3.3.5.10	pos_	14
3.3.5.11	scale_	14
3.3.5.12	sigma_	14
3.4	Format Struct Reference	15
3.4.1	Detailed Description	15
3.4.2	Member Data Documentation	15
3.4.2.1	start	15
3.4.2.2	step	15
3.4.2.3	stop	15
3.5	IOException Class Reference	15
3.5.1	Detailed Description	16
3.5.2	Constructor & Destructor Documentation	17
3.5.2.1	IOException	17
3.5.2.2	~IOException	17
3.5.3	Member Function Documentation	17
3.5.3.1	info	17
3.5.3.2	location	17
3.5.3.3	message	17
3.5.3.4	print	17
3.5.3.5	what	17
3.5.4	Member Data Documentation	18
3.5.4.1	location_	18
3.5.4.2	message_	18
3.6	Library Class Reference	18
3.6.1	Detailed Description	24
3.6.2	Constructor & Destructor Documentation	24
3.6.2.1	Library	24
3.6.2.2	Library	24
3.6.3	Member Function Documentation	24
3.6.3.1	add_base	24
3.6.3.2	add_convolute	24
3.6.3.3	add_curve	25
3.6.3.4	add_ending	25
3.6.3.5	add_format	25
3.6.3.6	add_geometry	25
3.6.3.7	add_scalar_name	26
3.6.3.8	add_scalars	26
3.6.3.9	add_scale	26

3.6.3.10	calculate_internals	26
3.6.3.11	calculate_partials	26
3.6.3.12	check_setup	26
3.6.3.13	complete_setup	27
3.6.3.14	from_binary	27
3.6.3.15	get_at	27
3.6.3.16	get_basis_name	27
3.6.3.17	get_central	28
3.6.3.18	get_convolute	28
3.6.3.19	get_ending	28
3.6.3.20	get_format	29
3.6.3.21	get_geo	29
3.6.3.22	get_internal	29
3.6.3.23	get_name	29
3.6.3.24	get_nbase	30
3.6.3.25	get_ncurves	30
3.6.3.26	get_nscalars	30
3.6.3.27	get_partial	30
3.6.3.28	get_scalar_at	31
3.6.3.29	get_scalar_name	31
3.6.3.30	get_scale	31
3.6.3.31	read_scale	32
3.6.3.32	set_atoms_info	32
3.6.3.33	set_atomtypes	32
3.6.3.34	set_name	32
3.6.3.35	set_nbase	33
3.6.3.36	set_ncurves	33
3.6.3.37	set_nscalars	33
3.6.3.38	to_binary	33
3.6.4	Friends And Related Function Documentation	33
3.6.4.1	Test.Library	33
3.6.5	Member Data Documentation	34
3.6.5.1	atoms_per_type_	34
3.6.5.2	atomtypes_	34
3.6.5.3	base_	34
3.6.5.4	central_mol_	34
3.6.5.5	central_molecule_	34
3.6.5.6	dimension_	34
3.6.5.7	endings_	34
3.6.5.8	geo_	34
3.6.5.9	index_matrix_	35
3.6.5.10	internals_	35
3.6.5.11	nadded_	35
3.6.5.12	nadded_scale_	35
3.6.5.13	name_	35
3.6.5.14	names_	35
3.6.5.15	natoms_	35
3.6.5.16	nbase_	36
3.6.5.17	ncurves_	36
3.6.5.18	no1_	36

3.6.5.19	no2_	36
3.6.5.20	no3_	36
3.6.5.21	no4_	36
3.6.5.22	npairs_	36
3.6.5.23	nscalars_	36
3.6.5.24	ntypes_	37
3.6.5.25	partials_	37
3.6.5.26	scalar_names_	37
3.6.5.27	scalars_	37
3.6.5.28	scale_	37
3.6.5.29	setup_done_	37
3.6.5.30	start_	37
3.6.5.31	step_	37
3.6.5.32	stop_	38
3.6.5.33	use_atoms_info_	38
3.6.5.34	version_	38
3.7	Matrix Class Reference	38
3.7.1	Detailed Description	39
3.7.2	Constructor & Destructor Documentation	39
3.7.2.1	Matrix	39
3.7.2.2	Matrix	39
3.7.3	Member Function Documentation	40
3.7.3.1	operator()	40
3.7.3.2	operator()	40
3.7.3.3	resize	40
3.7.4	Friends And Related Function Documentation	40
3.7.4.1	Test_Matrix	40
3.7.5	Member Data Documentation	41
3.7.5.1	columns_	41
3.7.5.2	data_	41
3.7.5.3	rows_	41
3.8	MeanScalarData Class Reference	41
3.8.1	Detailed Description	43
3.8.2	Constructor & Destructor Documentation	43
3.8.2.1	MeanScalarData	43
3.8.2.2	MeanScalarData	43
3.8.3	Member Function Documentation	44
3.8.3.1	accept	44
3.8.3.2	calculate_chi2	44
3.8.3.3	get_chi2	44
3.8.3.4	get_chi2_new	44
3.8.3.5	init	44
3.8.3.6	notify	45
3.8.3.7	print	45
3.8.3.8	weighted_analysis	45
3.8.4	Friends And Related Function Documentation	45
3.8.4.1	Test_MeanScalarData	45
3.8.5	Member Data Documentation	46
3.8.5.1	chi2_	46
3.8.5.2	chi2_new_	46

3.8.5.3	name_	46
3.8.5.4	nsample_	46
3.8.5.5	one_over_sigma2_	46
3.8.5.6	pos_	46
3.8.5.7	sigma_	46
3.8.5.8	target_	46
3.8.5.9	value_	47
3.8.5.10	value_new_	47
3.9	Mklib Class Reference	47
3.9.1	Detailed Description	48
3.9.2	Constructor & Destructor Documentation	48
3.9.2.1	Mklib	48
3.9.3	Member Function Documentation	48
3.9.3.1	check_keyword	48
3.9.3.2	compile_library	49
3.9.3.3	get_lib_version	49
3.9.3.4	library_from_info_file	49
3.9.4	Friends And Related Function Documentation	49
3.9.4.1	Test_Mklib	49
3.9.5	Member Data Documentation	50
3.9.5.1	debug_	50
3.9.5.2	verbos	50
3.10	Mlrnc Class Reference	50
3.10.1	Detailed Description	54
3.10.2	Constructor & Destructor Documentation	54
3.10.2.1	Mlrnc	54
3.10.3	Member Function Documentation	54
3.10.3.1	accept	54
3.10.3.2	first_print	54
3.10.3.3	get_chi2_new	54
3.10.3.4	init_chi2	54
3.10.3.5	last_print	54
3.10.3.6	montecarlo_test	55
3.10.3.7	move	55
3.10.3.8	notify	55
3.10.3.9	post_process	55
3.10.3.10	print	55
3.10.3.11	print_and_save	55
3.10.3.12	print_post_process_start	55
3.10.3.13	print_post_process_stop	55
3.10.3.14	read_ANALYSIS	56
3.10.3.15	read_CURVE	56
3.10.3.16	read_DISTRIBUTION	56
3.10.3.17	read_MEAN	56
3.10.3.18	read_PCF	56
3.10.3.19	read_RUN	56
3.10.3.20	read_SCALAR	56
3.10.3.21	read_section	57
3.10.3.22	read_VALUE	57
3.10.3.23	rmc_loop	57

3.10.3.24	run_rmc	57
3.10.3.25	save	57
3.10.4	Member Data Documentation	57
3.10.4.1	accepted_	57
3.10.4.2	accepted_recent_dump_	57
3.10.4.3	accepted_recent_print_	58
3.10.4.4	accepted_recent_probe_	58
3.10.4.5	analysis_chunk_size_	58
3.10.4.6	analysis_chunks_	58
3.10.4.7	ANALYSIS_section_	58
3.10.4.8	attempted_	58
3.10.4.9	attempted_recent_print_	58
3.10.4.10	chi2_	58
3.10.4.11	chi2_new_	59
3.10.4.12	delta_chi2_	59
3.10.4.13	dump_interval_	59
3.10.4.14	logfile_	59
3.10.4.15	moves_	59
3.10.4.16	print_interval_	59
3.10.4.17	probe_counter_	59
3.10.4.18	probe_interval_	59
3.10.4.19	rand_	60
3.10.4.20	RUN_section_	60
3.10.4.21	save_interval_	60
3.10.4.22	seed_	60
3.10.4.23	specswap_	60
3.10.4.24	title_	60
3.11	Mrnd Class Reference	60
3.11.1	Detailed Description	61
3.11.2	Constructor & Destructor Documentation	61
3.11.2.1	Mrnd	61
3.11.3	Member Function Documentation	61
3.11.3.1	random	61
3.11.3.2	set_seed	62
3.11.4	Friends And Related Function Documentation	62
3.11.4.1	Test.Mrnd	62
3.11.5	Member Data Documentation	62
3.11.5.1	seed_	62
3.12	PCFData Class Reference	62
3.12.1	Detailed Description	65
3.12.2	Constructor & Destructor Documentation	65
3.12.2.1	PCFData	65
3.12.2.2	PCFData	65
3.12.3	Member Function Documentation	66
3.12.3.1	accept	66
3.12.3.2	calculate_bin	66
3.12.3.3	calculate_chi2	66
3.12.3.4	calculate_partial_histogram	67
3.12.3.5	get_chi2	67
3.12.3.6	get_chi2_new	67

3.12.3.7	get_r	67
3.12.3.8	init	68
3.12.3.9	notify	68
3.12.3.10	print	68
3.12.3.11	print_to_file	68
3.12.3.12	read_reference	68
3.12.3.13	weighted_analysis	69
3.12.4	Friends And Related Function Documentation	69
3.12.4.1	Test_PCFData	69
3.12.5	Member Data Documentation	69
3.12.5.1	chi2_	69
3.12.5.2	chi2_new_	69
3.12.5.3	dr_	69
3.12.5.4	fit_interval_	70
3.12.5.5	histogram_	70
3.12.5.6	histogram_new_	70
3.12.5.7	nsample_	70
3.12.5.8	numberdensity_	70
3.12.5.9	one_over_dr_	70
3.12.5.10	one_over_sigma2_	70
3.12.5.11	partial_	70
3.12.5.12	pcf_normalization_factor_	71
3.12.5.13	pcf_reference_	71
3.12.5.14	rmax_	71
3.12.5.15	rmin_	71
3.12.5.16	sigma_	71
3.13	Sampleset Class Reference	71
3.13.1	Detailed Description	72
3.13.2	Constructor & Destructor Documentation	72
3.13.2.1	Sampleset	72
3.13.3	Member Function Documentation	73
3.13.3.1	add_index	73
3.13.3.2	get_indices	73
3.13.3.3	index_at	73
3.13.3.4	is_added	73
3.13.3.5	swap_into_slot	74
3.13.4	Friends And Related Function Documentation	74
3.13.4.1	Test_Sampleset	74
3.13.5	Member Data Documentation	74
3.13.5.1	indices_	74
3.13.5.2	nsample_	74
3.14	ScalarDistributionData Class Reference	74
3.14.1	Detailed Description	77
3.14.2	Constructor & Destructor Documentation	77
3.14.2.1	ScalarDistributionData	77
3.14.2.2	ScalarDistributionData	77
3.14.3	Member Function Documentation	77
3.14.3.1	accept	77
3.14.3.2	calculate_chi2	77
3.14.3.3	get_bin	78

3.14.3.4	get_chi2	78
3.14.3.5	get_chi2_new	78
3.14.3.6	init	78
3.14.3.7	notify	79
3.14.3.8	print	79
3.14.3.9	read_reference	79
3.14.3.10	weighted_analysis	79
3.14.4	Friends And Related Function Documentation	80
3.14.4.1	Test_ScalarDistributionData	80
3.14.5	Member Data Documentation	80
3.14.5.1	chi2_	80
3.14.5.2	chi2_new_	80
3.14.5.3	distribution_	80
3.14.5.4	distribution_new_	80
3.14.5.5	factor_	80
3.14.5.6	highest_	80
3.14.5.7	lowest_	81
3.14.5.8	name_	81
3.14.5.9	nbins_	81
3.14.5.10	nsample_	81
3.14.5.11	one_over_binsize_	81
3.14.5.12	one_over_sigma2_	81
3.14.5.13	pos_	81
3.14.5.14	scale_	81
3.14.5.15	sigma_	82
3.14.5.16	target_	82
3.15	Specswap Class Reference	82
3.15.1	Detailed Description	86
3.15.2	Constructor & Destructor Documentation	86
3.15.2.1	Specswap	86
3.15.2.2	Specswap	86
3.15.2.3	Specswap	87
3.15.3	Member Function Documentation	87
3.15.3.1	accept	87
3.15.3.2	add_curve	87
3.15.3.3	add_pcf	88
3.15.3.4	add_scalar_distribution	88
3.15.3.5	add_scalar_mean	88
3.15.3.6	add_scalar_value	89
3.15.3.7	analyse_chunk	89
3.15.3.8	calc_scalar_pos	89
3.15.3.9	chunk_analysis	90
3.15.3.10	collect_weights	90
3.15.3.11	get_chi2	90
3.15.3.12	get_chi2_new	90
3.15.3.13	move	90
3.15.3.14	notify	91
3.15.3.15	prepare_for_analysis	91
3.15.3.16	print	91
3.15.3.17	print_start	91

3.15.3.18	print_stop	91
3.15.3.19	print_to_file	92
3.15.3.20	print_weights	92
3.15.3.21	random	92
3.15.3.22	random_basis	92
3.15.3.23	setup	92
3.15.3.24	setup_chi2	93
3.15.3.25	setup_sampleset	93
3.15.3.26	weighted_analysis	93
3.15.3.27	write_dump	93
3.15.3.28	write_restart	93
3.15.3.29	write_weights_and_names_list	93
3.15.4	Friends And Related Function Documentation	94
3.15.4.1	Test_Specswap	94
3.15.5	Member Data Documentation	94
3.15.5.1	chi2_	94
3.15.5.2	chi2_new_	94
3.15.5.3	curve_data_	94
3.15.5.4	dump_counter_	94
3.15.5.5	from_basis_	94
3.15.5.6	from_sample_	94
3.15.5.7	library_	95
3.15.5.8	mean_scalar_data_	95
3.15.5.9	ncurves_	95
3.15.5.10	nprobe_	95
3.15.5.11	nsample_	95
3.15.5.12	pcf_data_	95
3.15.5.13	rand_	95
3.15.5.14	restart_	95
3.15.5.15	restart_path_	96
3.15.5.16	sampleset_	96
3.15.5.17	scalar_distribution_data_	96
3.15.5.18	slot_	96
3.15.5.19	value_scalar_data_	96
3.15.5.20	weights_	96
3.15.5.21	weights_table_	96
3.16	ValueScalarData Class Reference	97
3.16.1	Detailed Description	98
3.16.2	Constructor & Destructor Documentation	99
3.16.2.1	ValueScalarData	99
3.16.2.2	ValueScalarData	99
3.16.3	Member Function Documentation	99
3.16.3.1	accept	99
3.16.3.2	calculate_chi2	99
3.16.3.3	get_chi2	99
3.16.3.4	get_chi2_new	100
3.16.3.5	init	100
3.16.3.6	notify	100
3.16.3.7	print	100
3.16.3.8	weighted_analysis	101

3.16.4	Friends And Related Function Documentation	101
3.16.4.1	Test_ValueScalarData	101
3.16.5	Member Data Documentation	101
3.16.5.1	chi2_	101
3.16.5.2	chi2_new_	101
3.16.5.3	interval_	101
3.16.5.4	name_	101
3.16.5.5	nsample_	102
3.16.5.6	one_over_sigma2_	102
3.16.5.7	pos_	102
3.16.5.8	sigma_	102
3.16.5.9	target_	102
3.16.5.10	value_	102
3.16.5.11	value_new_	102
4	File Documentation	103
4.1	basiscontainer.h File Reference	103
4.1.1	Detailed Description	103
4.2	curvedata.cpp File Reference	103
4.2.1	Detailed Description	104
4.3	curvedata.h File Reference	104
4.3.1	Detailed Description	104
4.4	ioexception.cpp File Reference	104
4.4.1	Detailed Description	104
4.5	ioexception.h File Reference	104
4.5.1	Detailed Description	105
4.6	ioutils.cpp File Reference	105
4.6.1	Detailed Description	107
4.6.2	Function Documentation	107
4.6.2.1	check_eof	107
4.6.2.2	check_path	107
4.6.2.3	check_positive_integer	108
4.6.2.4	check_sigma	108
4.6.2.5	eatline	108
4.6.2.6	empty_file_error	108
4.6.2.7	eof	109
4.6.2.8	error	109
4.6.2.9	error_exit	109
4.6.2.10	missing_keyword_error	109
4.6.2.11	newline_indent	110
4.6.2.12	open_file_error	110
4.6.2.13	print_startup	110
4.6.2.14	print_success	110
4.6.2.15	read_keyword_error	110
4.6.2.16	same_keyword_error	111
4.6.2.17	same_section_error	111
4.6.2.18	start_timer	111
4.6.2.19	timer	111
4.6.2.20	timestamp	111
4.6.2.21	to_string	112

4.6.2.22	unknown_keyword_error	112
4.6.2.23	unknown_section_error	112
4.6.3	Variable Documentation	112
4.6.3.1	time_0__	112
4.7	ioutils.h File Reference	112
4.7.1	Detailed Description	115
4.7.2	Define Documentation	115
4.7.2.1	FUNCTION	115
4.7.2.2	LOCATION	115
4.7.3	Function Documentation	115
4.7.3.1	check_eof	115
4.7.3.2	check_path	115
4.7.3.3	check_positive_integer	116
4.7.3.4	check_sigma	116
4.7.3.5	eatline	116
4.7.3.6	empty_file_error	116
4.7.3.7	eof	116
4.7.3.8	error	117
4.7.3.9	error_exit	117
4.7.3.10	missing_keyword_error	117
4.7.3.11	newline_indent	117
4.7.3.12	open_file_error	118
4.7.3.13	print_startup	118
4.7.3.14	print_success	118
4.7.3.15	read_keyword_error	118
4.7.3.16	same_keyword_error	118
4.7.3.17	same_section_error	119
4.7.3.18	start_timer	119
4.7.3.19	timer	119
4.7.3.20	timestamp	119
4.7.3.21	to_string	119
4.7.3.22	unknown_keyword_error	120
4.7.3.23	unknown_section_error	120
4.8	library.cpp File Reference	120
4.8.1	Detailed Description	121
4.9	library.h File Reference	121
4.9.1	Detailed Description	121
4.10	mathutils.cpp File Reference	121
4.10.1	Detailed Description	123
4.10.2	Function Documentation	123
4.10.2.1	calculate_distance	123
4.10.2.2	operator*	123
4.10.2.3	operator*	124
4.10.2.4	operator+	124
4.10.2.5	operator-	124
4.10.2.6	operator-	125
4.10.2.7	operator/	125
4.10.2.8	setup_index_matrix	125
4.10.2.9	vadd	126
4.10.2.10	vadd	126

4.10.2.11	vnormalize	126
4.10.2.12	vsquare	126
4.10.2.13	vsub	127
4.10.2.14	vsub	127
4.10.2.15	vsum	127
4.10.2.16	vsum	127
4.11	mathutils.h File Reference	128
4.11.1	Detailed Description	129
4.11.2	Function Documentation	129
4.11.2.1	calculate_distance	129
4.11.2.2	fastfloor	130
4.11.2.3	operator*	130
4.11.2.4	operator*	130
4.11.2.5	operator+	131
4.11.2.6	operator-	131
4.11.2.7	operator-	131
4.11.2.8	operator/	132
4.11.2.9	setup_index_matrix	132
4.11.2.10	vadd	132
4.11.2.11	vadd	132
4.11.2.12	vnormalize	133
4.11.2.13	vsquare	133
4.11.2.14	vsub	133
4.11.2.15	vsub	133
4.11.2.16	vsum	134
4.11.2.17	vsum	134
4.11.3	Variable Documentation	134
4.11.3.1	PI_	134
4.12	matrix.cpp File Reference	134
4.12.1	Detailed Description	134
4.13	matrix.h File Reference	135
4.13.1	Detailed Description	135
4.14	meanscalardata.cpp File Reference	135
4.14.1	Detailed Description	135
4.15	meanscalardata.h File Reference	135
4.15.1	Detailed Description	136
4.16	mklb.cpp File Reference	136
4.16.1	Detailed Description	136
4.17	mklb.h File Reference	136
4.17.1	Detailed Description	137
4.18	mklbmain.cpp File Reference	137
4.18.1	Detailed Description	137
4.18.2	Function Documentation	137
4.18.2.1	main	137
4.18.2.2	print_help	138
4.18.2.3	print_version	138
4.19	mlrmc.cpp File Reference	138
4.19.1	Detailed Description	138
4.20	mlrmc.h File Reference	138
4.20.1	Detailed Description	139

4.21	mlrnd.cpp File Reference	139
4.21.1	Detailed Description	139
4.22	mlrnd.h File Reference	139
4.22.1	Detailed Description	139
4.23	pcfdata.cpp File Reference	140
4.23.1	Detailed Description	140
4.24	pcfdata.h File Reference	140
4.24.1	Detailed Description	140
4.25	randf.f File Reference	140
4.25.1	Function Documentation	141
4.25.1.1	RND1	141
4.26	randf.h File Reference	141
4.26.1	Detailed Description	141
4.26.2	Function Documentation	141
4.26.2.1	rnd1_	141
4.27	sampleset.cpp File Reference	141
4.27.1	Detailed Description	142
4.28	sampleset.h File Reference	142
4.28.1	Detailed Description	142
4.29	scalardistributiondata.cpp File Reference	142
4.29.1	Detailed Description	142
4.30	scalardistributiondata.h File Reference	143
4.30.1	Detailed Description	143
4.31	specswap.cpp File Reference	143
4.31.1	Detailed Description	143
4.32	specswap.h File Reference	143
4.32.1	Detailed Description	144
4.33	specswapmain.cpp File Reference	144
4.33.1	Detailed Description	145
4.33.2	Function Documentation	145
4.33.2.1	main	145
4.34	valuescalardata.cpp File Reference	145
4.34.1	Detailed Description	145
4.35	valuescalardata.h File Reference	145
4.35.1	Detailed Description	146

Chapter 1

Class Index

1.1 Class List

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

BasisContainer (A minimal struct to bunch together an index, a weight and a basis name)	5
Convolute (Struct for encapsulating the convolute parameters)	7
CurveData (Class representing a curve data set)	8
Format (Struct for encapsulating the format parameters)	15
IOException (Class for representing an IO exception)	15
Library (The class defining the library to use in SpecSwap-RMC)	18
Matrix (Class for representing a 2D double matrix)	38
MeanScalarData (Class for representing a mean scalar data set)	41
Mklib (Class for implementing the main functionality of the mklib program) . .	47
Mlrnc (The central RMC driver class, handling input setup and program flow)	50
Mlrnd (Class for representing the random number generator)	60
PCFData (Class representing a PCF data set)	62
Sampleset (Class representing the Sampleset in the SpecSwap simulation, for book keeping indices and performing swap moves)	71
ScalarDistributionData (Class for representing a scalar distribution data set) .	74
Specswap (The central Specswap workhorse object performing moves and controlling communication with attached data sets)	82
ValueScalarData (Class representing a value scalar data set)	97

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

basiscontainer.h (File for the BasisContainer definition and implementation)	103
curvedata.cpp (File for the CurveData class implementation)	103
curvedata.h (File for the CurveData class definition)	104
ioexception.cpp (File for the IOException class implementation)	104
ioexception.h (File for the IOException class definition)	104
ioutils.cpp (File containing implementations for functions defined in ioutils.h)	105
ioutils.h (File containing definitions of macros and utility functions for IO and error handling)	112
library.cpp (File for the Library class definition)	120
library.h (File for the Library class definition)	121
mathutils.cpp (File containing implementations of the utility functions defined in mathutils.h)	121
mathutils.h (File containing definitions of utility functions and constants for mathematical operation)	128
matrix.cpp (File for the Matrix class implementation)	134
matrix.h (File for the Matrix class definition)	135
meanscalardata.cpp (File for the MeanScalarData class definition)	135
meanscalardata.h (File for the MeanScalarData class definition)	135
mklb.cpp (File for the Mklb class implementation)	136
mklb.h (File for the Mklb class definition)	136
mklbmain.cpp (File for the implementation of a simple command line interface to the Mklb library compiler utility)	137
mlrmc.cpp (File for the Mlrmc class implementation)	138
mlrmc.h (File for the Mlrmc class definition)	138
mlrnd.cpp (File for the Mlrnd random number generator class implementation)	139
mlrnd.h (File for the Mlrnd random number generator class definition)	139
pcfdata.cpp (File for the PCFData class implementation)	140
pcfdata.h (File for the PCFData class definition)	140
randf.f	140

randf.h (File for a C declaration of the FORTRAN rnd1 routine)	141
sampleset.cpp (File for the Sampleset class implementation)	141
sampleset.h (File for the Sampleset class definition)	142
scalardistributiondata.cpp (File for the ScalarDistributionData class implemen- tation)	142
scalardistributiondata.h (File for the ScalarDistributionData class definition) . .	143
specswap.cpp (File for the Specswap class implementation)	143
specswap.h (File for the Specswap class definition)	143
specswapmain.cpp (File for the SpecSwap program's main routine)	144
valuescalardata.cpp (File for the ValueScalarData class implementation) . . .	145
valuescalardata.h (File for the ValueScalarData class definition)	145

Chapter 3

Class Documentation

3.1 BasisContainer Struct Reference

A minimal struct to bunch together an index, a weight and a basis name.

```
#include <basiscontainer.h>
```

Static Public Member Functions

- static bool [compareIndex](#) (const [BasisContainer](#) &first, const [BasisContainer](#) &second)

Compare function to sort according to lowest index.

- static bool [compareWeight](#) (const [BasisContainer](#) &first, const [BasisContainer](#) &second)

Compare function to sort according to highest weight.

Public Attributes

- int [index](#)

The index.

- double [weight](#)

The weight.

- std::string [name](#)

The name.

3.1.1 Detailed Description

A minimal struct to bunch together an index, a weight and a basis name.

Definition at line 26 of file basiscontainer.h.

3.1.2 Member Function Documentation

3.1.2.1 `static bool BasisContainer::compareIndex (const BasisContainer & first, const BasisContainer & second)` `[inline, static]`

Compare function to sort according to lowest index.

Definition at line 34 of file basiscontainer.h.

3.1.2.2 `static bool BasisContainer::compareWeight (const BasisContainer & first, const BasisContainer & second)` `[inline, static]`

Compare function to sort according to highest weight.

Definition at line 36 of file basiscontainer.h.

3.1.3 Member Data Documentation

3.1.3.1 `int BasisContainer::index`

The index.

Definition at line 28 of file basiscontainer.h.

3.1.3.2 `std::string BasisContainer::name`

The name.

Definition at line 32 of file basiscontainer.h.

3.1.3.3 `double BasisContainer::weight`

The weight.

Definition at line 30 of file basiscontainer.h.

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

- [basiscontainer.h](#)

3.2 Convolute Struct Reference

Struct for encapsulating the convolute parameters.

```
#include <library.h>
```

Public Attributes

- int [no1](#)
- int [no2](#)
- int [no3](#)
- int [no4](#)

3.2.1 Detailed Description

Struct for encapsulating the convolute parameters.

Definition at line 38 of file library.h.

3.2.2 Member Data Documentation

3.2.2.1 int Convolute::no1

Definition at line 39 of file library.h.

3.2.2.2 int Convolute::no2

Definition at line 40 of file library.h.

3.2.2.3 int Convolute::no3

Definition at line 41 of file library.h.

3.2.2.4 int Convolute::no4

Definition at line 42 of file library.h.

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

- [library.h](#)

3.3 CurveData Class Reference

Class representing a curve data set.

```
#include <curvedata.h>
```

Public Member Functions

- [CurveData](#) ()
Default constructor for the [CurveData](#) class needed for initializing from the [Specswap](#) class.
- [CurveData](#) (const double sigma, const bool area_renorm, const std::string &curve_name, const std::string &path, const [Library](#) &library)
Constructor for the [CurveData](#) class.
- void [init](#) (const std::vector< int > &sampleset, const [Library](#) &library)
Function for setting up the curve and calculating initial chi2.
- void [notify](#) (const int from_sample, const int from_basis, const [Library](#) &library)
Function to notify the curve object of an attempted move.
- void [accept](#) ()
Function for indicating that the move should be accepted, which means chi2 should be set to chi2 new, and the data should be updated.
- double [get_chi2](#) () const
Query for the chi2 value.
- double [get_chi2_new](#) () const
Query for the chi2 value.
- void [print_to_file](#) (const std::string &basename) const
Function for printing the curve and reference to file.
- void [print](#) () const
Print chi2 info to screen.
- void [weighted_analysis](#) (const [Library](#) &library, const std::vector< [BasisContainer](#) > &weights_table, const std::string &basename) const
Print weighted and unweighted data.

Private Member Functions

- void [read_reference](#) (const std::string &path)
Helper routine for reading the reference data from the input file.
- const double [calculate_chi2](#) (const std::vector< double > &curve)
Helper routine for calculating chi2 for a given curve.

Private Attributes

- int [nsample_](#)
The number of elements in the sample set. Needed for normalization.
- double [sigma_](#)
The sigma value to use for calculating chi2.
- double [one_over_sigma2_](#)
One over sigma squared, for convenience.
- double [chi2_](#)
The chi2 value.
- double [chi2_new_](#)
The chi2 value for the attempted move.
- int [pos_](#)
The position in the library this curve corresponds to.
- bool [area_renorm_](#)
Flag indicating if the curve should be area renormalized.
- std::string [curve_name_](#)
The curve name (in the library).
- std::vector< double > [scale_](#)
Holding the scale for the curve.
- std::vector< double > [curve_reference_](#)
Holding the reference curve data from the data file.
- std::vector< double > [curve_](#)
Holding the current curve.
- std::vector< double > [curve_new_](#)
Holding the curve for the attempted move.

Friends

- class [Test_CurveData](#)

Declaring the testclass as friend to facilitate testing.

3.3.1 Detailed Description

Class representing a curve data set.

Definition at line 34 of file curvedata.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 `CurveData::CurveData ()` `[inline]`

Default constructor for the [CurveData](#) class needed for initializing from the [Specswap](#) class.

Definition at line 41 of file curvedata.h.

3.3.2.2 `CurveData::CurveData (const double sigma, const bool area_renorm, const std::string & curve_name, const std::string & path, const Library & library)`

Constructor for the [CurveData](#) class.

Parameters

<i>sigma</i>	: The sigma value to use for the fit.
<i>curve_name</i>	: The name (ending) of the curve.
<i>path</i>	: The path to the file with reference data.
<i>library</i>	: The library to use in the rmc run.

Definition at line 33 of file curvedata.cpp.

3.3.3 Member Function Documentation

3.3.3.1 `void CurveData::accept ()`

Function for indicating that the move should be accepted, which means chi2 should be set to chi2 new, and the data should be updated.

Definition at line 204 of file curvedata.cpp.

3.3.3.2 `const double CurveData::calculate_chi2 (const std::vector< double > & curve)`
[private]

Helper routine for calculating chi2 for a given curve.

Parameters

<i>curve</i>	: The curve to calculate for.
--------------	-------------------------------

Returns

: The corresponding chi2

Definition at line 124 of file curvedata.cpp.

3.3.3.3 `double CurveData::get_chi2 () const` [inline]

Query for the chi2 value.

Returns

: chi2 .

Definition at line 79 of file curvedata.h.

3.3.3.4 `double CurveData::get_chi2_new () const` [inline]

Query for the chi2 value.

Returns

: chi2 .

Definition at line 84 of file curvedata.h.

3.3.3.5 `void CurveData::init (const std::vector< int > & sampleset, const Library & library)`

Function for setting up the curve and calculating initial chi2.

Parameters

<i>sampleset</i>	: Vector holding the sample set indices.
<i>library</i>	: The library to take the data from.

Definition at line 154 of file curvedata.cpp.

3.3.3.6 void CurveData::notify (const int *from_sample*, const int *from_basis*, const Library & *library*)

Function to notify the curve object of an attempted move.

Parameters

<i>from_sample</i>	: The global index of the basis element to swap out of the sampleset.
<i>from_basis</i>	: The global index of the basis element to swap into the sampleset.
<i>library</i>	: The library to take the data from.

Definition at line 179 of file curvedata.cpp.

3.3.3.7 void CurveData::print () const

Print chi2 info to screen.

Definition at line 218 of file curvedata.cpp.

3.3.3.8 void CurveData::print_to_file (const std::string & *basename*) const

Function for printing the curve and reference to file.

Parameters

<i>basename</i>	: The base name to save under.
-----------------	--------------------------------

Definition at line 227 of file curvedata.cpp.

3.3.3.9 void CurveData::read_reference (const std::string & *path*) [private]

Helper routine for reading the reference data from the input file.

Parameters

<i>path</i>	: The full path to the file to read.
-------------	--------------------------------------

Definition at line 92 of file curvedata.cpp.

3.3.3.10 void CurveData::weighted_analysis (const Library & *library*, const std::vector< BasisContainer > & *weights_table*, const std::string & *basename*) const

Print weighted and unweighted data.

Parameters

<i>library</i>	: The library to use.
----------------	-----------------------

<i>weights_ - table</i>	: The list of basis elements with weights and names.
<i>basename</i>	: The base file name to write to.

Definition at line 258 of file `curvedata.cpp`.

3.3.4 Friends And Related Function Documentation

3.3.4.1 friend class Test_CurveData [friend]

Declaring the testclass as friend to facilitate testing.

Definition at line 156 of file `curvedata.h`.

3.3.5 Member Data Documentation

3.3.5.1 bool CurveData::area_renorm_ [private]

Flag indicating if the curve should be area renormalized.

Definition at line 138 of file `curvedata.h`.

3.3.5.2 double CurveData::chi2_ [private]

The chi2 value.

Definition at line 129 of file `curvedata.h`.

3.3.5.3 double CurveData::chi2_new_ [private]

The chi2 value for the attempted move.

Definition at line 132 of file `curvedata.h`.

3.3.5.4 std::vector<double> CurveData::curve_ [private]

Holding the current curve.

Definition at line 150 of file `curvedata.h`.

3.3.5.5 std::string CurveData::curve_name_ [private]

The curve name (in the library).

Definition at line 141 of file `curvedata.h`.

3.3.5.6 `std::vector<double> CurveData::curve_new_` [private]

Holding the curve for the attempted move.

Definition at line 153 of file `curvedata.h`.

3.3.5.7 `std::vector<double> CurveData::curve_reference_` [private]

Holding the reference curve data from the data file.

Definition at line 147 of file `curvedata.h`.

3.3.5.8 `int CurveData::nsample_` [private]

The number of elements in the sample set. Needed for normalization.

Definition at line 120 of file `curvedata.h`.

3.3.5.9 `double CurveData::one_over_sigma2_` [private]

One over sigma squared, for convenience.

Definition at line 126 of file `curvedata.h`.

3.3.5.10 `int CurveData::pos_` [private]

The position in the library this curve corresponds to.

Definition at line 135 of file `curvedata.h`.

3.3.5.11 `std::vector<double> CurveData::scale_` [private]

Holding the scale for the curve.

Definition at line 144 of file `curvedata.h`.

3.3.5.12 `double CurveData::sigma_` [private]

The sigma value to use for calculating chi2.

Definition at line 123 of file `curvedata.h`.

The documentation for this class was generated from the following files:

- [curvedata.h](#)
- [curvedata.cpp](#)

3.4 Format Struct Reference

Struct for encapsulating the format parameters.

```
#include <library.h>
```

Public Attributes

- int [start](#)
- int [stop](#)
- int [step](#)

3.4.1 Detailed Description

Struct for encapsulating the format parameters.

Definition at line 46 of file library.h.

3.4.2 Member Data Documentation

3.4.2.1 int Format::start

Definition at line 47 of file library.h.

3.4.2.2 int Format::step

Definition at line 49 of file library.h.

3.4.2.3 int Format::stop

Definition at line 48 of file library.h.

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

- [library.h](#)

3.5 IOException Class Reference

Class for representing an IO exception.

```
#include <ioexception.h>
```

Public Member Functions

- [IOException](#) (const std::string &message, const std::string &location)
Constructor.
- virtual [~IOException](#) () throw ()
Implementing the destructor.
- virtual const char * [what](#) () const throw ()
Overriding the base class [what\(\)](#) function.
- void [print](#) () const
Function for printing an error message.
- std::string [message](#) () const
Query function for the message.
- std::string [location](#) () const
Query function for the location.

Private Member Functions

- std::string [info](#) () const
Function for generating the crash info.

Private Attributes

- std::string [message_](#)
The error message.
- std::string [location_](#)
The error location.

3.5.1 Detailed Description

Class for representing an IO exception.

Definition at line 27 of file ioexception.h.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 IOException::IOException (const std::string & *message*, const std::string & *location*)

Constructor.

Parameters

<i>message</i>	: The error message.
<i>location</i>	: The error location.

Definition at line 29 of file `ioexception.cpp`.

3.5.2.2 virtual IOException::~IOException () throw () [inline, virtual]

Implementing the destructor.

Definition at line 39 of file `ioexception.h`.

3.5.3 Member Function Documentation

3.5.3.1 std::string IOException::info () const [private]

Function for generating the crash info.

Definition at line 57 of file `ioexception.cpp`.

3.5.3.2 std::string IOException::location () const [inline]

Query function for the location.

Definition at line 55 of file `ioexception.h`.

3.5.3.3 std::string IOException::message () const [inline]

Query function for the message.

Definition at line 51 of file `ioexception.h`.

3.5.3.4 void IOException::print () const

Function for printing an error message.

Definition at line 49 of file `ioexception.cpp`.

3.5.3.5 const char * IOException::what () const throw () [virtual]

Overriding the base class `what()` function.

Definition at line 41 of file `ioexception.cpp`.

3.5.4 Member Data Documentation

3.5.4.1 `std::string IOException::location_` [private]

The error location.

Definition at line 67 of file `ioexception.h`.

3.5.4.2 `std::string IOException::message_` [private]

The error message.

Definition at line 65 of file `ioexception.h`.

The documentation for this class was generated from the following files:

- [ioexception.h](#)
- [ioexception.cpp](#)

3.6 Library Class Reference

The class defining the library to use in SpecSwap-RMC.

```
#include <library.h>
```

Public Member Functions

- [Library](#) ()
Default constructor.
- [Library](#) (const std::string &filename)
Costructor for construction from file.
- void [set_name](#) (const std::string &name)
For setting the name of the library.
- void [set_ncurves](#) (const int ncurves)
To set the number of curves.
- void [add_convolute](#) (const double no1, const double no2, const double no3, const double no4)
For adding convolute parameters.
- void [add_format](#) (const double start, const double stop, const double step)

For adding format parameters.

- void [add_scale](#) (const std::string &filename)
For adding a scale.
- void [add_ending](#) (const std::string &ending)
For adding an ending.
- void [set_nscalars](#) (const int nscalars)
For setting the number of scalars.
- void [add_scalar_name](#) (const std::string &name)
For adding a scalar name.
- void [set_atomtypes](#) (const std::vector< std::string > &atomtypes)
To set the atom types of the system. Must be the same for all basis elements.
- void [set_atoms_info](#) (const std::vector< int > &atoms_per_type, const std::vector< int > ¢ral_molecule)
To set the atoms information for all basis elements. If this is called it overrides any information about atoms per type and central molecule stored in the xyz files when generating a basis using [Mklib](#).
- void [set_nbase](#) (const int nbase)
To set the number of base elements to fill the library with.
- void [add_base](#) (const std::string &basename)
Adding a base, given its filename, reading curves, scalars and geometry from the file with the given base name. and calculating the correlation functions.
- void [complete_setup](#) ()
Function for finalizing setup.
- const int [get_nbase](#) () const
Query function for the number of basis elements.
- const int [get_ncurves](#) () const
Query function for the number of curves for each basis element.
- const int [get_nscalars](#) () const
Query function for the number of scalars for each basis element.
- const std::string & [get_ending](#) (const int pos) const
Query function for the ending of a curve.
- const std::string & [get_scalar_name](#) (const int pos) const
Query function for the name of a scalar.

- [Convolute](#) [get_convolute](#) (const int pos) const
Query function for the convolute parameters of a curve.
- [Format](#) [get_format](#) (const int pos) const
Query function for the format parameters of a curve.
- const std::string [get_name](#) () const
Query function for the library name.
- const std::string & [get_basis_name](#) (const int index) const
Query function for the name of a basis element.
- const std::vector< double > & [get_partial](#) (const int index, const int type1, const int type2) const
Query function for a partial distance distribution of a basis.
- const std::vector< double > & [get_internal](#) (const int index, const int type1, const int type2) const
Query function for the central molecule internal distance distribution of a basis.
- const std::vector< int > & [get_central](#) (const int index) const
Query function for a central molecule of a basis element.
- const [Matrix](#) & [get_geo](#) (int index) const
Query function for the geometry matrix of a basis.
- const std::vector< double > & [get_at](#) (int index, int pos) const
Query function for the values of a basis curve.
- std::vector< double > [get_scale](#) (const int pos) const
Query function for the scale of a curve.
- const double [get_scalar_at](#) (const int index, const int pos) const
Query function for the value of a scalar for a certain basis function.
- void [to_binary](#) (const std::string &filename) const
Utility function to write a file to binary format.

Static Public Attributes

- static const std::string [version_](#) = "version-3.0"
The version string of the library.

Private Member Functions

- void `check_setup` (const std::string &function, const std::string &location="") const

Raises an error with the message saying the function_name is not callable after setup has been completed.

- void `read_scale` (const std::string &filename)

Read a scale from file.

- void `add_curve` (const std::string &filename, const int pos)

Add a curve from file.

- void `add_scalars` (const std::string &filename)

Add a set of scalars to a basis.

- void `add_geometry` (const std::string &filename)

Add the geometry of a basis element.

- void `calculate_partials` ()

Calculate and save the partial distance distributions for the last added geometry.

- void `calculate_internals` ()

Calculate and save the internal (molecular) distance distributions for the last added geometry.

- void `from_binary` (const std::string &filename)

Utility function to setup from a binary file.

Private Attributes

- std::vector< std::vector< std::vector< double > > > `base_`

The vector of basis functions.

- std::vector< std::vector< std::vector< double > > > `partials_`

The vector of partials.

- std::vector< std::vector< std::vector< double > > > `internals_`

The vector of internal geometries.

- std::vector< `Matrix` > `geo_`

The vector of geometries.

- std::vector< std::vector< int > > `central_mol_`

The indices vector of the central molecule.

- `std::vector< std::vector< int > > natoms_`
The number of atoms per basis. This is needed for normalizing the partials.
- `std::vector< std::vector< double > > scalars_`
The scalars data.
- `std::vector< std::string > names_`
The vector of base-names of the basis functions.
- `std::vector< std::string > endings_`
The name ending of the curves.
- `std::vector< std::vector< double > > scale_`
The scales of the curves.
- `std::vector< double > no1_`
The no1 parameter.
- `std::vector< double > no2_`
The no2 parameter.
- `std::vector< double > no3_`
The no3 parameter.
- `std::vector< double > no4_`
The no4 parameter.
- `std::vector< double > start_`
The start parameter.
- `std::vector< double > stop_`
The stop parameter.
- `std::vector< double > step_`
The step parameter.
- `std::vector< int > dimension_`
The dimension parameter.
- `std::vector< std::string > scalar_names_`
The names of the scalars.
- `std::vector< std::string > atomtypes_`
The names of the atom types.

- int [nadded_scale_](#)
counter to hold the current position when adding scales.
- int [nadded_](#)
Holds the number of added basis.
- int [nbase_](#)
The total number of bases.
- int [ntypes_](#)
The number of atom types.
- int [npairs_](#)
The number of pairs.
- int [ncurves_](#)
The number of curves.
- int [nscalars_](#)
The number of scalars.
- bool [use_atoms_info_](#)
Flag for indicating that we have read general atoms per type and central molecule info.
- bool [setup_done_](#)
Flag indicating if the setup is finished.
- std::vector< int > [atoms_per_type_](#)
Storage of general atoms per type info to use for all basis elements.
- std::vector< int > [central_molecule_](#)
Storage of general central molecule info to use for all basis elements.
- std::string [name_](#)
The name of the library.
- [Matrix index_matrix_](#)
Data structure for easy conversion between indices and partials.

Friends

- class [Test_Library](#)
Declare the test class as friend to facilitate testing.

3.6.1 Detailed Description

The class defining the library to use in SpecSwap-RMC.

Definition at line 54 of file library.h.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Library::Library ()

Default constructor.

Definition at line 47 of file library.cpp.

3.6.2.2 Library::Library (const std::string & filename)

Costructor for construction from file.

Parameters

<i>filename</i>	: The name of the binary library file to read from.
-----------------	---

Definition at line 64 of file library.cpp.

3.6.3 Member Function Documentation

3.6.3.1 void Library::add_base (const std::string & basename)

Adding a base, given its filename, reading curves, scalars and geometry from the file with the given base name. and calculating the correlation functions.

Parameters

<i>basename</i>	: The base-name of the base to add.
-----------------	-------------------------------------

Definition at line 215 of file library.cpp.

3.6.3.2 void Library::add_convolute (const double no1, const double no2, const double no3, const double no4)

For adding convolute parameters.

Parameters

<i>no1</i>	: The no1 parameter.
<i>no2</i>	: The no2 parameter.
<i>no3</i>	: The no3 parameter.
<i>no4</i>	: The no4 parameter.

Definition at line 105 of file library.cpp.

3.6.3.3 void Library::add_curve (const std::string & *filename*, const int *pos*) [private]

Add a curve from file.

Parameters

<i>filename</i>	The full name of the file to read.
<i>pos</i>	The position index to which the part should be added. (Needed for knowing the dimensions.)

Definition at line 272 of file library.cpp.

3.6.3.4 void Library::add_ending (const std::string & *ending*)

For adding an ending.

Parameters

<i>ending</i>	: The curve ending (name) to add.
---------------	-----------------------------------

Definition at line 146 of file library.cpp.

3.6.3.5 void Library::add_format (const double *start*, const double *stop*, const double *step*)

For adding format parameters.

Parameters

<i>start</i>	: The start value of the curve.
<i>stop</i>	: The stop value of the curve.
<i>step</i>	: The step value of the curve.

Definition at line 121 of file library.cpp.

3.6.3.6 void Library::add_geometry (const std::string & *filename*) [private]

Add the geometry of a basis element.

Parameters

<i>filename</i>	The full name of the geometry (xyz) file to read.
-----------------	---

Definition at line 361 of file library.cpp.

3.6.3.7 void Library::add_scalar_name (const std::string & name)

For adding a scalar name.

Parameters

<i>name</i>	: The name of the scalar to add.
-------------	----------------------------------

Definition at line 166 of file library.cpp.

3.6.3.8 void Library::add_scalars (const std::string & filename) [private]

Add a set of scalars to a basis.

Parameters

<i>filename</i>	The full name of the scalars file to read.
-----------------	--

Definition at line 335 of file library.cpp.

3.6.3.9 void Library::add_scale (const std::string & filename)

For adding a scale.

Parameters

<i>filename</i>	The file name to read the scale data from.
-----------------	--

Definition at line 136 of file library.cpp.

3.6.3.10 void Library::calculate_internals () [private]

Calculate and save the internal (molecular) distance distributions for the last added geometry.

Definition at line 577 of file library.cpp.

3.6.3.11 void Library::calculate_partials () [private]

Calculate and save the partial distance distributions for the last added geometry.

Definition at line 484 of file library.cpp.

3.6.3.12 void Library::check_setup (const std::string & function, const std::string & location = " ") const [private]

Raises an error with the message saying the function_name is not callable after setup has been completed.

Parameters

<i>function</i>	: The name of the function to appear in the error message.
<i>location</i>	: The location of the error.

Definition at line 1357 of file library.cpp.

3.6.3.13 void Library::complete_setup ()

Function for finalizing setup.

Definition at line 258 of file library.cpp.

3.6.3.14 void Library::from_binary (const std::string & filename) [private]

Utility function to setup from a binary file.

Parameters

<i>filename</i>	The full path of the file to read from, without the .library ending.
-----------------	--

Definition at line 952 of file library.cpp.

3.6.3.15 const std::vector<double>& Library::get_at (int index, int pos) const [inline]

Query function for the values of a basis curve.

Parameters

<i>index</i>	The index number of the basis element.
<i>pos</i>	The curve position number.

Returns

The curve data.

Definition at line 275 of file library.h.

3.6.3.16 const std::string& Library::get_basis_name (const int index) const [inline]

Query function for the name of a basis element.

Parameters

<i>index</i>	: The index of the basis element.
--------------	-----------------------------------

Returns

The name.

Definition at line 234 of file library.h.

3.6.3.17 `const std::vector<int>& Library::get_central (const int index) const` `[inline]`

Query function for a central molecule of a basis element.

Parameters

<i>index</i>	: The index of the basis element.
--------------	-----------------------------------

Returns

The list of indices for the central molecule.

Definition at line 258 of file library.h.

3.6.3.18 `Convolute Library::get_convolute (const int pos) const` `[inline]`

Query function for the convolute parameters of a curve.

Parameters

<i>pos</i>	: The position number of the curve.
------------	-------------------------------------

Returns

The name convolute parameters of the curve.

Definition at line 202 of file library.h.

3.6.3.19 `const std::string& Library::get_ending (const int pos) const` `[inline]`

Query function for the ending of a curve.

Parameters

<i>pos</i>	: The position number of the curve.
------------	-------------------------------------

Returns

The name ending of the curve.

Definition at line 186 of file library.h.

3.6.3.20 Format Library::get_format (const int *pos*) const `[inline]`

Query function for the format parameters of a curve.

Parameters

<i>pos</i>	: The position number of the curve.
------------	-------------------------------------

Returns

The format parameters of the curve.

Definition at line 215 of file library.h.

3.6.3.21 const Matrix& Library::get_geo (int *index*) const `[inline]`

Query function for the geometry matrix of a basis.

Parameters

<i>index</i>	: The index of the basis element.
--------------	-----------------------------------

Returns

The coordinate matrix of the basis element.

Definition at line 266 of file library.h.

3.6.3.22 const std::vector< double > & Library::get_internal (const int *index*, const int *type1*, const int *type2*) const

Query function for the central molecule internal distance distribution of a basis.

Parameters

<i>index</i>	: The index of the basis function.
<i>type1</i>	: The first atom type.
<i>type2</i>	: The second atom type.

Returns

The list of intra molecular distances between atoms of type1 and type2 in basis index.

Definition at line 1341 of file library.cpp.

3.6.3.23 const std::string Library::get_name () const `[inline]`

Query function for the library name.

Returns

The name.

Definition at line 226 of file library.h.

3.6.3.24 const int Library::get_nbase () const [inline]

Query function for the number of basis elements.

Returns

The number of added basis functions.

Definition at line 164 of file library.h.

3.6.3.25 const int Library::get_ncurves () const [inline]

Query function for the number of curves for each basis element.

Returns

The number of curves.

Definition at line 171 of file library.h.

3.6.3.26 const int Library::get_nscalars () const [inline]

Query function for the number of scalars for each basis element.

Returns

The number of scalars.

Definition at line 178 of file library.h.

3.6.3.27 const std::vector< double > & Library::get_partial (const int *index*, const int *type1*, const int *type2*) const

Query function for a partial distance distribution of a basis.

Parameters

<i>index</i>	: The index of the basis function.
<i>type1</i>	: The first atom type.
<i>type2</i>	: The second atom type.

Returns

The list of distances between atoms of type1 and type2 in basis index.

Definition at line 1332 of file library.cpp.

3.6.3.28 `const double Library::get_scalar_at (const int index, const int pos) const`
`[inline]`

Query function for the value of a scalar for a certain basis function.

Parameters

<i>index</i>	The index of the basis function.
<i>pos</i>	The possition number of the scalar.

Returns

The value of the scalar.

Definition at line 292 of file library.h.

3.6.3.29 `const std::string& Library::get_scalar_name (const int pos) const` `[inline]`

Query function for the name of a scalar.

Parameters

<i>pos</i>	: The possition number of the scalar.
------------	---------------------------------------

Returns

The name of the scalar.

Definition at line 194 of file library.h.

3.6.3.30 `std::vector<double> Library::get_scale (const int pos) const` `[inline]`

Query function for the scale of a curve.

Parameters

<i>pos</i>	: The possition number of the curve.
------------	--------------------------------------

Returns

The scale (x-values) for the desiered curve.

Definition at line 283 of file library.h.

3.6.3.31 void Library::read_scale (const std::string & *filename*) [private]

Read a scale from file.

Parameters

<i>filename</i>	The full name of the file to read.
-----------------	------------------------------------

Definition at line 658 of file library.cpp.

3.6.3.32 void Library::set_atoms_info (const std::vector< int > & *atoms_per_type*, const std::vector< int > & *central_molecule*)

To set the atoms information for all basis elements. If this is called it overrides any information about atoms per type and central molecule stored in the xyz files when generating a basis using [Mklib](#).

Parameters

<i>atoms_per_type</i>	: The vector of atoms per type, with the first element being the total number of atoms.
<i>central_molecule</i>	: The vector holding the indices of the central molecule.

Definition at line 190 of file library.cpp.

3.6.3.33 void Library::set_atomtypes (const std::vector< std::string > & *atomtypes*)

To set the atom types of the system. Must be the same for all basis elements.

Parameters

<i>atomtypes</i>	The vector of atomtypes.
------------------	--------------------------

Definition at line 176 of file library.cpp.

3.6.3.34 void Library::set_name (const std::string & *name*)

For setting the name of the library.

Parameters

<i>name</i>	: The name.
-------------	-------------

Definition at line 85 of file library.cpp.

3.6.3.35 void Library::set_nbase (const int *nbase*)

To set the number of base elements to fill the library with.

Parameters

<i>nbase</i>	: The number of base elements.
--------------	--------------------------------

Definition at line 204 of file library.cpp.

3.6.3.36 void Library::set_ncurves (const int *ncurves*)

To set the number of curves.

Parameters

<i>ncurves</i>	: The number of curves.
----------------	-------------------------

Definition at line 95 of file library.cpp.

3.6.3.37 void Library::set_nscalars (const int *nscalars*)

For setting the number of scalars.

Parameters

<i>nscalars</i>	The number of scalars.
-----------------	------------------------

Definition at line 156 of file library.cpp.

3.6.3.38 void Library::to_binary (const std::string & *filename*) const

Utility function to write a file to binary format.

Parameters

<i>filename</i>	The name of the file to write to.
-----------------	-----------------------------------

Definition at line 723 of file library.cpp.

3.6.4 Friends And Related Function Documentation**3.6.4.1 friend class Test_Library [friend]**

Declare the test class as friend to facilitate testing.

Definition at line 460 of file library.h.

3.6.5 Member Data Documentation

3.6.5.1 `std::vector<int> Library::atoms_per_type_` [private]

Storage of general atoms per type info to use for all basis elements.

Definition at line 451 of file library.h.

3.6.5.2 `std::vector<std::string> Library::atomtypes_` [private]

The names of the atom types.

Definition at line 419 of file library.h.

3.6.5.3 `std::vector< std::vector < std::vector<double> > > Library::base_` [private]

The vector of basis functions.

Definition at line 368 of file library.h.

3.6.5.4 `std::vector< std::vector<int> > Library::central_mol_` [private]

The indices vector of the central molecule.

Definition at line 376 of file library.h.

3.6.5.5 `std::vector<int> Library::central_molecule_` [private]

Storage of general central molecule info to use for all basis elements.

Definition at line 453 of file library.h.

3.6.5.6 `std::vector<int> Library::dimension_` [private]

The dimension parameter.

Definition at line 407 of file library.h.

3.6.5.7 `std::vector< std::string > Library::endings_` [private]

The name ending of the curves.

Definition at line 388 of file library.h.

3.6.5.8 `std::vector<Matrix> Library::geo_` [private]

The vector of geometries.

Definition at line 374 of file library.h.

3.6.5.9 Matrix Library::index_matrix_ [private]

Data structure for easy conversion between indices an partials.

Definition at line 457 of file library.h.

3.6.5.10 std::vector< std::vector< std::vector<double> > > Library::internals_ [private]

The vector of internal geometries.

Definition at line 372 of file library.h.

3.6.5.11 int Library::nadded_ [private]

Holds the number of added basis.

Definition at line 427 of file library.h.

3.6.5.12 int Library::nadded_scale_ [private]

counter to hold the current possition when adding scales.

Definition at line 425 of file library.h.

3.6.5.13 std::string Library::name_ [private]

The name of the library.

Definition at line 455 of file library.h.

3.6.5.14 std::vector< std::string > Library::names_ [private]

The vector of base-names of the basis functions.

Definition at line 382 of file library.h.

3.6.5.15 std::vector< std::vector<int> > Library::natoms_ [private]

The number of atoms per basis. This is needed for normalizing the partials.

Definition at line 378 of file library.h.

3.6.5.16 `int Library::nbase_ [private]`

The total number of bases.

Definition at line 429 of file library.h.

3.6.5.17 `int Library::ncurves_ [private]`

The number of curves.

Definition at line 439 of file library.h.

3.6.5.18 `std::vector<double> Library::no1_ [private]`

The no1 parameter.

Definition at line 393 of file library.h.

3.6.5.19 `std::vector<double> Library::no2_ [private]`

The no2 parameter.

Definition at line 395 of file library.h.

3.6.5.20 `std::vector<double> Library::no3_ [private]`

The no3 parameter.

Definition at line 397 of file library.h.

3.6.5.21 `std::vector<double> Library::no4_ [private]`

The no4 parameter.

Definition at line 399 of file library.h.

3.6.5.22 `int Library::npairs_ [private]`

The number of pairs.

Definition at line 437 of file library.h.

3.6.5.23 `int Library::nscalars_ [private]`

The number of scalars.

Definition at line 441 of file library.h.

3.6.5.24 `int Library::ntypes_` [private]

The number of atom types.

Definition at line 435 of file library.h.

3.6.5.25 `std::vector< std::vector< std::vector<double> > > Library::partials_`
[private]

The vector of partials.

Definition at line 370 of file library.h.

3.6.5.26 `std::vector< std::string > Library::scalar_names_` [private]

The names of the scalars.

Definition at line 413 of file library.h.

3.6.5.27 `std::vector< std::vector<double> > Library::scalars_` [private]

The scalars data.

Definition at line 380 of file library.h.

3.6.5.28 `std::vector< std::vector<double> > Library::scale_` [private]

The scales of the curves.

Definition at line 390 of file library.h.

3.6.5.29 `bool Library::setup_done_` [private]

Flag indicating if the setup is finished.

Definition at line 449 of file library.h.

3.6.5.30 `std::vector<double> Library::start_` [private]

The start parameter.

Definition at line 401 of file library.h.

3.6.5.31 `std::vector<double> Library::step_` [private]

The step parameter.

Definition at line 405 of file library.h.

3.6.5.32 `std::vector<double> Library::stop_` [private]

The stop parameter.

Definition at line 403 of file library.h.

3.6.5.33 `bool Library::use_atoms_info_` [private]

Flag for indicating that we have read general atoms per type and central molecule info.

Definition at line 447 of file library.h.

3.6.5.34 `const std::string Library::version_ = "version-3.0"` [static]

The version string of the library.

Definition at line 309 of file library.h.

The documentation for this class was generated from the following files:

- [library.h](#)
- [library.cpp](#)

3.7 Matrix Class Reference

Class for representing a 2D double matrix.

```
#include <matrix.h>
```

Public Member Functions

- [Matrix](#) ()
Default constructor.
- [Matrix](#) (int rows, int columns)
Constructor.
- void [resize](#) (int rows, int columns)
Function for resizing the matrix.
- double & [operator\(\)](#) (int row, int column)
Access operator.
- double [operator\(\)](#) (int row, int column) const
Access operator (const version).

Private Attributes

- int `rows_`
The number of rows.
- int `columns_`
The number of columns.
- `std::vector< std::vector< double > >` `data_`
The matrix data.

Friends

- class `Test_Matrix`
Declaring the test class as friend to facilitate testing.

3.7.1 Detailed Description

Class for representing a 2D double matrix.

Definition at line 29 of file `matrix.h`.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 `Matrix::Matrix ()`

Default constructor.

Definition at line 27 of file `matrix.cpp`.

3.7.2.2 `Matrix::Matrix (int rows, int columns)`

Constructor.

Parameters

<i>rows</i>	: The number of rows in the matrix.
<i>columns</i>	: The number of columns in the matrix.

Definition at line 37 of file `matrix.cpp`.

3.7.3 Member Function Documentation

3.7.3.1 `double& Matrix::operator()(int row, int column)` `[inline]`

Access operator.

Parameters

<i>row</i>	: The row to access.
<i>column</i>	: The column to access.

Definition at line 53 of file matrix.h.

3.7.3.2 `double Matrix::operator()(int row, int column) const` `[inline]`

Access operator (const version).

Parameters

<i>row</i>	: The row to access.
<i>column</i>	: The column to access.

Definition at line 59 of file matrix.h.

3.7.3.3 `void Matrix::resize(int rows, int columns)`

Function for resizing the matrix.

Parameters

<i>rows</i>	: The number of rows in the matrix.
<i>columns</i>	: The number of columns in the matrix.

Definition at line 48 of file matrix.cpp.

3.7.4 Friends And Related Function Documentation

3.7.4.1 `friend class Test_Matrix` `[friend]`

Declaring the test class as friend to facilitate testing.

Definition at line 73 of file matrix.h.

3.7.5 Member Data Documentation

3.7.5.1 `int Matrix::columns_` `[private]`

The number of columns.

Definition at line 68 of file `matrix.h`.

3.7.5.2 `std::vector< std::vector<double> > Matrix::data_` `[private]`

The matrix data.

Definition at line 70 of file `matrix.h`.

3.7.5.3 `int Matrix::rows_` `[private]`

The number of rows.

Definition at line 66 of file `matrix.h`.

The documentation for this class was generated from the following files:

- [matrix.h](#)
- [matrix.cpp](#)

3.8 MeanScalarData Class Reference

Class for representing a mean scalar data set.

```
#include <meanscalardata.h>
```

Public Member Functions

- [MeanScalarData](#) ()
Default constructor needed for initialization from specswap.
- [MeanScalarData](#) (const std::string &name, const double target, const double sigma, const int pos)
Constructor.
- void [init](#) (const std::vector< int > &sampleset, const [Library](#) &library)
Function for setting up the initial mean value and initial chi2.
- void [notify](#) (const int from_sample, const int from_basis, const [Library](#) &library)
Function to notify of an attempted move.
- void [accept](#) ()

Function for indicating that the move should be accepted, which means chi2 should be set to chi2 new, and the data should be updated.

- double `get_chi2 ()` const
Query for the chi2 value.
- double `get_chi2_new ()` const
Query for the chi2 value.
- void `print ()` const
Print chi2 info to screen.
- void `weighted_analysis` (const `Library` &library, const std::vector< `BasisContainer` > &weights_table, const std::string &basename) const
Print weighted and unweighted data.

Private Member Functions

- double `calculate_chi2` (const double value) const
Helper routine to calculate chi2.

Private Attributes

- std::string `name_`
The name (ending) of the scalar.
- double `target_`
The target value.
- double `sigma_`
The sigma value.
- double `one_over_sigma2_`
The one over sigma2.
- double `value_`
The value.
- double `value_new_`
The value for the attempted move.
- double `chi2_`
The chi2 value.

- double [chi2_new_](#)
The chi2 value of the attempted move.
- int [pos_](#)
The index of the scalar in the library.
- int [nsample_](#)
The number of elements in the sample set.

Friends

- class [Test_MeanScalarData](#)
Declaring the test class as friend to facilitate testing.

3.8.1 Detailed Description

Class for representing a mean scalar data set.

Definition at line 34 of file meanscalardata.h.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 MeanScalarData::MeanScalarData () [inline]

Default constructor needed for initialization from specswap.

Definition at line 40 of file meanscalardata.h.

3.8.2.2 MeanScalarData::MeanScalarData (const std::string & *name*, const double *target*, const double *sigma*, const int *pos*)

Constructor.

Parameters

<i>name</i>	: The name (ending) of the scalar.
<i>target</i>	: Target mean value.
<i>sigma</i>	: The sigma value.
<i>pos</i>	: The index of the scalar in the library.

Definition at line 33 of file meanscalardata.cpp.

3.8.3 Member Function Documentation

3.8.3.1 void MeanScalarData::accept ()

Function for indicating that the move should be accepted, which means chi2 should be set to chi2 new, and the data should be updated.

Definition at line 100 of file meanscalardata.cpp.

3.8.3.2 double MeanScalarData::calculate_chi2 (const double *value*) const [private]

Helper routine to calculate chi2.

Parameters

<i>value</i>	: The value to calculate for.
--------------	-------------------------------

Returns

: chi2

Definition at line 48 of file meanscalardata.cpp.

3.8.3.3 double MeanScalarData::get_chi2 () const [inline]

Query for the chi2 value.

Returns

: chi2 .

Definition at line 77 of file meanscalardata.h.

3.8.3.4 double MeanScalarData::get_chi2_new () const [inline]

Query for the chi2 value.

Returns

: chi2 .

Definition at line 82 of file meanscalardata.h.

3.8.3.5 void MeanScalarData::init (const std::vector< int > & *sampleset*, const Library & *library*)

Function for setting up the initial mean value and initial chi2.

Parameters

<i>sampleset</i>	: Vector holding the sample set indices.
<i>library</i>	: The library to take the data from.

Definition at line 56 of file meanscalardata.cpp.

3.8.3.6 void MeanScalarData::notify (const int *from_sample*, const int *from_basis*, const Library & *library*)

Function to notify of an attempted move.

Parameters

<i>from_samle</i>	: The global index of the basis element to swap out of the sampleset.
<i>from_basis</i>	: The global index of the basis element to swap into the sampleset.
<i>library</i>	: The library to take the data from.

Definition at line 82 of file meanscalardata.cpp.

3.8.3.7 void MeanScalarData::print () const

Print chi2 info to screen.

Definition at line 108 of file meanscalardata.cpp.

3.8.3.8 void MeanScalarData::weighted_analysis (const Library & *library*, const std::vector< BasisContainer > & *weights_table*, const std::string & *basename*) const

Print weighted and unweighted data.

Parameters

<i>library</i>	: The library to use.
<i>weights_table</i>	: The list of basis elements with weights and names.
<i>basename</i>	: The base file name to write to.

Definition at line 119 of file meanscalardata.cpp.

3.8.4 Friends And Related Function Documentation

3.8.4.1 friend class Test_MeanScalarData [friend]

Declaring the test class as friend to facilitate testing.

Definition at line 129 of file meanscalardata.h.

3.8.5 Member Data Documentation

3.8.5.1 `double MeanScalarData::chi2_` [private]

The chi2 value.

Definition at line 120 of file meanscalardata.h.

3.8.5.2 `double MeanScalarData::chi2_new_` [private]

The chi2 value of the attempted move.

Definition at line 122 of file meanscalardata.h.

3.8.5.3 `std::string MeanScalarData::name_` [private]

The name (ending) of the scalar.

Definition at line 108 of file meanscalardata.h.

3.8.5.4 `int MeanScalarData::nsample_` [private]

The number of elements in the sample set.

Definition at line 126 of file meanscalardata.h.

3.8.5.5 `double MeanScalarData::one_over_sigma2_` [private]

The one over sigma2.

Definition at line 114 of file meanscalardata.h.

3.8.5.6 `int MeanScalarData::pos_` [private]

The index of the scalar in the library.

Definition at line 124 of file meanscalardata.h.

3.8.5.7 `double MeanScalarData::sigma_` [private]

The sigma value.

Definition at line 112 of file meanscalardata.h.

3.8.5.8 `double MeanScalarData::target_` [private]

The target value.

Definition at line 110 of file meanscalardata.h.

3.8.5.9 double MeanScalarData::value_ [private]

The value.

Definition at line 116 of file meanscalardata.h.

3.8.5.10 double MeanScalarData::value_new_ [private]

The value for the attempted move.

Definition at line 118 of file meanscalardata.h.

The documentation for this class was generated from the following files:

- [meanscalardata.h](#)
- [meanscalardata.cpp](#)

3.9 Mklib Class Reference

Class for implementing the main functionality of the mklib program.

```
#include <mklib.h>
```

Public Member Functions

- [Mklib](#) (const bool debug=false)
- void [compile_library](#) (const std::string &basename)

Function for compiling a new binary library file from specifications in a .info file.

Static Public Member Functions

- static const std::string [get_lib_version](#) ()

Static function to return the library version string.

Private Member Functions

- [Library library_from_info_file](#) (const std::string &basename, const std::string &debugname="NONAME")

The workhorse function for setting up a library from a .info file.

- void `check_keyword` (const std::string &key, const std::string &found, const std::string &filename, const std::string &location)

Helper routine to check keyword errors and throw an error if needed.

Private Attributes

- bool `debug_`

Flag indicating if we run in debug/test mode. With this flag set to true no calls are made to functions that reads data from file, other than the main input file. Insead the function call is printed to standard out.

Static Private Attributes

- static const bool `verbos` = false

Flag for indicating verbos printout for debugging.

Friends

- class `Test_Mklib`

Declare the test class a friend to facilitate testing.

3.9.1 Detailed Description

Class for implementing the main functionality of the mklib program.

Definition at line 30 of file mklib.h.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 Mklib::Mklib (const bool `debug` = false)

Constructor providing a way to set the debug flag for testing.

Definition at line 34 of file mklib.cpp.

3.9.3 Member Function Documentation

3.9.3.1 void Mklib::check_keyword (const std::string & `key`, const std::string & `found`, const std::string & `filename`, const std::string & `location`) [private]

Helper routine to check keyword errors and throw an error if needed.

Parameters

<i>key</i>	: The keyword to check.
<i>found</i>	: The keyword found.
<i>filename</i>	: The name of the file we were reading from.
<i>location</i>	: The the location of the check.

Definition at line 356 of file mklib.cpp.

3.9.3.2 void Mklib::compile_library (const std::string & *basename*)

Function for compiling a new binary library file from specifications in a .info file.

Parameters

<i>basename</i>	: The base name of the the .info file to construct from.
-----------------	--

Definition at line 41 of file mklib.cpp.

3.9.3.3 static const std::string Mklib::get_lib_version () [inline, static]

Static function to return the library version string.

Definition at line 47 of file mklib.h.

3.9.3.4 Library Mklib::library_from_info_file (const std::string & *basename*, const std::string & *debugname* = "NONAME") [private]

The workhorse function for setting up a library from a .info file.

Parameters

<i>basename</i>	: The folder in which the .info file with the same name to read is to be found.
<i>debugname</i>	: Name for debugging. Takes a full path.

Definition at line 62 of file mklib.cpp.

3.9.4 Friends And Related Function Documentation**3.9.4.1 friend class Test_Mklib [friend]**

Declare the test class a friend to facilitate testing.

Definition at line 84 of file mklib.h.

3.9.5 Member Data Documentation

3.9.5.1 `bool Mklib::debug_` `[private]`

Flag indicating if we run in debug/test mode. With this flag set to true no calls are made to functions that reads data from file, other than the main input file. Insead the function call is printed to standard out.

Definition at line 81 of file mklib.h.

3.9.5.2 `const bool Mklib::verbos = false` `[static, private]`

Flag for indicating verbos printout for debugging.

Definition at line 74 of file mklib.h.

The documentation for this class was generated from the following files:

- [mklib.h](#)
- [mklib.cpp](#)

3.10 Mlrnc Class Reference

The central RMC driver class, handling input setup and program flow.

```
#include <mlrnc.h>
```

Public Member Functions

- [Mlrnc](#) (std::string filename)
The constructor for the RMC object.
- void [run_rmc](#) ()
- void [post_process](#) ()
Performs post processing.

Public Attributes

- [Mlrnd rand_](#)
the random number generator.

Private Member Functions

- void [read_section](#) (const std::string section_key, std::ifstream &infile)

Function for reading an input section keyword.

- void [read_RUN](#) (std::ifstream &infile)
Reading a RUN input section. infile : The open input filestream.
- void [read_PCF](#) (std::ifstream &infile)
Reading a PCF input section. infile : The open input filestream.
- void [read_CURVE](#) (std::ifstream &infile)
Reading CURVE input. infile : The open input filestream.
- void [read_SCALAR](#) (std::ifstream &infile)
Reading SCALAR input. infile : The open input filestream.
- void [read_MEAN](#) (std::ifstream &infile, const std::string &scalarname)
Reading MEAN input. infile : The open input filestream.
- void [read_VALUE](#) (std::ifstream &infile, const std::string &scalarname)
Reading VALUE input. infile : The open input filestream.
- void [read_DISTRIBUTION](#) (std::ifstream &infile, const std::string &scalarname)
Reading DISTRIBUTION input. infile : The open input filestream.
- void [read_ANALYSIS](#) (std::ifstream &infile)
Reading an ANALYSIS input section. infile : The open input filestream.
- void [rmc_loop](#) ()
The main RMC loop.
- void [init_chi2](#) ()
The collection of Chi2 from datasets.
- void [move](#) ()
Make a move.
- void [notify](#) ()
Notify the datasets that a move has been made.
- void [get_chi2_new](#) ()
Collect the new Chi2 from the data sets.
- bool [montecarlo_test](#) () const
Perform the MC test.
- void [accept](#) ()
Accept the move.

- void `print_and_save` ()
Handle all IO during run.
- void `print` ()
Handle screen and chi2 file IO.
- void `first_print` ()
Prints some starting information to screen.
- void `last_print` ()
Prints some ending information to screen.
- void `save` () const
IO function for saving a configuration to file.
- void `print_post_process_start` () const
Prints some post processing start information to screen.
- void `print_post_process_stop` () const
Prints some post processing stop information to screen.

Private Attributes

- `Specswap specsswap_`
The main `Specswap` object performing specsmap moves and interfacing the library.
- bool `RUN_section_`
Flag for indicating if the RUN input section has been read.
- bool `ANALYSIS_section_`
Flag for indicating if the ANALYSIS input section has been read.
- int `seed_`
The seed to the random number generator.
- int `print_interval_`
Parameter for holding how often to print to screen.
- int `save_interval_`
Parameter for holding how often to save to file.
- int `probe_interval_`
Parameter for holding the probe interval.

- int [dump_interval_](#)
Parameter for holding the dump interval.
- int [analysis_chunks_](#)
Holds number of chunks in analysis.
- int [analysis_chunk_size_](#)
Holds the size of the chunks in analysis.
- unsigned long int [moves_](#)
Parameter holding the number of moves the simulation should run.
- unsigned long int [probe_counter_](#)
Counter holding the number of times the structure was probed.
- unsigned long int [accepted_](#)
The number of accepted moves.
- unsigned long int [attempted_](#)
The number of attempted moves.
- int [accepted_recent_print_](#)
The number of accepted since last print.
- int [accepted_recent_probe_](#)
The number of accepted since last probe.
- int [accepted_recent_dump_](#)
The number of accepted moves since last dump.
- int [attempted_recent_print_](#)
The number of attempted moves since last print.
- double [chi2_](#)
The chi2.
- double [chi2_new_](#)
The new chi2.
- double [delta_chi2_](#)
The difference in chi2.
- std::string [title_](#)
- std::ofstream [logfile_](#)
The logfile.

3.10.1 Detailed Description

The central RMC driver class, handling input setup and program flow.

Definition at line 38 of file mlrmc.h.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 `Mrmc::Mrmc (std::string filename)`

The constructor for the RMC object.

Parameters

<i>filename</i>	: The basename of the inputfile.
-----------------	----------------------------------

Definition at line 38 of file mlrmc.cpp.

3.10.3 Member Function Documentation

3.10.3.1 `void Mrmc::accept () [private]`

Accept the move.

Definition at line 1167 of file mlrmc.cpp.

3.10.3.2 `void Mrmc::first_print () [private]`

Prints some starting information to screen.

Definition at line 1250 of file mlrmc.cpp.

3.10.3.3 `void Mrmc::get_chi2_new () [private]`

Collect the new Chi2 from the data sets.

Definition at line 1135 of file mlrmc.cpp.

3.10.3.4 `void Mrmc::init_chi2 () [private]`

The collection of Chi2 from datasets.

Definition at line 1107 of file mlrmc.cpp.

3.10.3.5 `void Mrmc::last_print () [private]`

Prints some ending information to screen.

Definition at line 1272 of file mlrnc.cpp.

3.10.3.6 `bool Mlrnc::montecarlo_test () const` [private]

Perform the MC test.

Definition at line 1145 of file mlrnc.cpp.

3.10.3.7 `void Mlrnc::move ()` [private]

Make a move.

Definition at line 1117 of file mlrnc.cpp.

3.10.3.8 `void Mlrnc::notify ()` [private]

Notify the datasets that a move has been made.

Definition at line 1126 of file mlrnc.cpp.

3.10.3.9 `void Mlrnc::post_process ()`

Performs post processing.

Definition at line 1374 of file mlrnc.cpp.

3.10.3.10 `void Mlrnc::print ()` [private]

Handle screen and chi2 file IO.

Definition at line 1222 of file mlrnc.cpp.

3.10.3.11 `void Mlrnc::print_and_save ()` [private]

Handle all IO during run.

Definition at line 1185 of file mlrnc.cpp.

3.10.3.12 `void Mlrnc::print_post_process_start () const` [private]

Prints some post processing start information to screen.

Definition at line 1287 of file mlrnc.cpp.

3.10.3.13 `void Mlrnc::print_post_process_stop () const` [private]

Prints some post processing stop information to screen.

Definition at line 1300 of file mlrmc.cpp.

3.10.3.14 void Mlrmc::read_ANALYSIS (std::ifstream & *infile*) [private]

Reading an ANALYSIS input section. *infile* : The open input filestream.

Definition at line 1055 of file mlrmc.cpp.

3.10.3.15 void Mlrmc::read_CURVE (std::ifstream & *infile*) [private]

Reading CURVE input. *infile* : The open input filestream.

Definition at line 756 of file mlrmc.cpp.

3.10.3.16 void Mlrmc::read_DISTRIBUTION (std::ifstream & *infile*, const std::string & *scalarname*) [private]

Reading DISTRIBUTION input. *infile* : The open input filestream.

Definition at line 1019 of file mlrmc.cpp.

3.10.3.17 void Mlrmc::read_MEAN (std::ifstream & *infile*, const std::string & *scalarname*) [private]

Reading MEAN input. *infile* : The open input filestream.

Definition at line 922 of file mlrmc.cpp.

3.10.3.18 void Mlrmc::read_PCF (std::ifstream & *infile*) [private]

Reading a PCF input section. *infile* : The open input filestream.

Definition at line 498 of file mlrmc.cpp.

3.10.3.19 void Mlrmc::read_RUN (std::ifstream & *infile*) [private]

Reading a RUN input section. *infile* : The open input filestream.

Definition at line 183 of file mlrmc.cpp.

3.10.3.20 void Mlrmc::read_SCALAR (std::ifstream & *infile*) [private]

Reading SCALAR input. *infile* : The open input filestream.

Definition at line 873 of file mlrmc.cpp.

3.10.3.21 `void Mlrnc::read_section (const std::string section_key, std::ifstream & infile)`
`[private]`

Function for reading an input section keyword.

Parameters

<code><i>section_key</i></code>	: The section keyword read.	<code><i>infile</i></code>	: The open input filestream.
---------------------------------	-----------------------------	----------------------------	------------------------------

Definition at line 138 of file `mlrnc.cpp`.

3.10.3.22 `void Mlrnc::read_VALUE (std::ifstream & infile, const std::string & scalarname)`
`[private]`

Reading VALUE input. `infile` : The open input filestream.

Definition at line 960 of file `mlrnc.cpp`.

3.10.3.23 `void Mlrnc::rmc_loop ()` `[private]`

The main RMC loop.

Definition at line 1342 of file `mlrnc.cpp`.

3.10.3.24 `void Mlrnc::run_rmc ()`

/brief Function for triggering the rmc simulation to start.

Definition at line 1309 of file `mlrnc.cpp`.

3.10.3.25 `void Mlrnc::save () const` `[private]`

IO function for saving a configuration to file.

3.10.4 Member Data Documentation

3.10.4.1 `unsigned long int Mlrnc::accepted_` `[private]`

The number of accepted moves.

Definition at line 228 of file `mlrnc.h`.

3.10.4.2 `int Mlrnc::accepted_recent_dump_` `[private]`

The number of accepted moves since last dump.

Definition at line 236 of file `mlrnc.h`.

3.10.4.3 int Mlrmc::accepted_recent_print_ [private]

The number of accepted since last print.

Definition at line 232 of file mlrmc.h.

3.10.4.4 int Mlrmc::accepted_recent_probe_ [private]

The number of accepted since last probe.

Definition at line 234 of file mlrmc.h.

3.10.4.5 int Mlrmc::analysis_chunk_size_ [private]

Holds the size of the chunks in analysis.

Definition at line 217 of file mlrmc.h.

3.10.4.6 int Mlrmc::analysis_chunks_ [private]

Holds number of chunks in analysis.

Definition at line 215 of file mlrmc.h.

3.10.4.7 bool Mlrmc::ANALYSIS_section_ [private]

Flag for indicating if the ANALYSIS input section has been read.

Definition at line 198 of file mlrmc.h.

3.10.4.8 unsigned long int Mlrmc::attempted_ [private]

The number of attempted moves.

Definition at line 230 of file mlrmc.h.

3.10.4.9 int Mlrmc::attempted_recent_print_ [private]

The number of attempted moves since last print.

Definition at line 238 of file mlrmc.h.

3.10.4.10 double Mlrmc::chi2_ [private]

The chi2.

Definition at line 245 of file mlrmc.h.

3.10.4.11 `double Mlrnc::chi2_new_ [private]`

The new chi2.

Definition at line 247 of file mlrnc.h.

3.10.4.12 `double Mlrnc::delta_chi2_ [private]`

The difference in chi2.

Definition at line 249 of file mlrnc.h.

3.10.4.13 `int Mlrnc::dump_interval_ [private]`

Parameter for holding the dump interval.

Definition at line 213 of file mlrnc.h.

3.10.4.14 `std::ofstream Mlrnc::logfile_ [private]`

The logfile.

Definition at line 258 of file mlrnc.h.

3.10.4.15 `unsigned long int Mlrnc::moves_ [private]`

Parameter holding the number of moves the simulation should run.

Definition at line 219 of file mlrnc.h.

3.10.4.16 `int Mlrnc::print_interval_ [private]`

Parameter for holding how often to print to screen.

Definition at line 207 of file mlrnc.h.

3.10.4.17 `unsigned long int Mlrnc::probe_counter_ [private]`

Counter holding the number of times the structure was probed.

Definition at line 226 of file mlrnc.h.

3.10.4.18 `int Mlrnc::probe_interval_ [private]`

Parameter for holding the probe interval.

Definition at line 211 of file mlrnc.h.

3.10.4.19 `Mrnd Mlrmc::rand_`

the random number generator.

Definition at line 63 of file `mlrmc.h`.

3.10.4.20 `bool Mlrmc::RUN_section_` `[private]`

Flag for indicating if the RUN input section has been read.

Definition at line 196 of file `mlrmc.h`.

3.10.4.21 `int Mlrmc::save_interval_` `[private]`

Parameter for holding how often to save to file.

Definition at line 209 of file `mlrmc.h`.

3.10.4.22 `int Mlrmc::seed_` `[private]`

The seed to the random number generator.

Definition at line 205 of file `mlrmc.h`.

3.10.4.23 `Specswap Mlrmc::specswap_` `[private]`

The main [Specswap](#) object performing specswap moves and interfacing the library.

Definition at line 189 of file `mlrmc.h`.

3.10.4.24 `std::string Mlrmc::title_` `[private]`

Definition at line 256 of file `mlrmc.h`.

The documentation for this class was generated from the following files:

- [mlrmc.h](#)
- [mlrmc.cpp](#)

3.11 Mrnd Class Reference

Class for representing the random number generator.

```
#include <mlrnd.h>
```

Public Member Functions

- [Mlrnd](#) ()
Default constructor.
- void [set_seed](#) (int seed)
Function for setting the seed value.
- double [random](#) () const
Function for getting the next random number.

Private Attributes

- int [seed_](#)
The seed value.

Friends

- class [Test_Mlrnd](#)
Declaring the test class as friend to facilitate testing.

3.11.1 Detailed Description

Class for representing the random number generator.
Definition at line 27 of file mlrnd.h.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 Mlrnd::Mlrnd ()

Default constructor.
Definition at line 31 of file mlrnd.cpp.

3.11.3 Member Function Documentation

3.11.3.1 double Mlrnd::random () const

Function for getting the next random number.

Note

: This function calls the 'static' fortran backend.

Returns

: A pseudo random number between 0 and 1.

Definition at line 47 of file mlrnd.cpp.

3.11.3.2 void Mlrnd::set_seed (int seed)

Function for setting the seed value.

Parameters

<i>seed</i>	: The seed value to use.
-------------	--------------------------

Definition at line 39 of file mlrnd.cpp.

3.11.4 Friends And Related Function Documentation**3.11.4.1 friend class Test_Mlrnd [friend]**

Declaring the test class as friend to facilitate testing.

Definition at line 52 of file mlrnd.h.

3.11.5 Member Data Documentation**3.11.5.1 int Mlrnd::seed_ [private]**

The seed value.

Definition at line 49 of file mlrnd.h.

The documentation for this class was generated from the following files:

- [mlrnd.h](#)
- [mlrnd.cpp](#)

3.12 PCFData Class Reference

Class representing a PCF data set.

```
#include <pcfdata.h>
```

Public Member Functions

- [PCFData](#) ()
Default constructor for the [PCFData](#) class needed for initializing from the [Specswap](#) class.
- [PCFData](#) (const double rmin, const double rmax, const double dr, const double sigma, const double numberdensity, const std::pair< double, double > &fit_interval, const int nbins, const std::pair< int, int > &partial, const std::string &path)
Constructor for the [PCFData](#) class.
- void [init](#) (const std::vector< int > &sampleset, const [Library](#) &library)
Function for setting up the pcf and calculating initial chi2.
- void [notify](#) (const int from_sample, const int from_basis, const [Library](#) &library)
Function to notify the pcfdata object of an attempted move.
- void [accept](#) ()
Function for indicating that the move should be accepted, which means chi2 should be set to chi2 new, and the data should be updated.
- double [get_chi2](#) () const
Query for the chi2 value.
- double [get_chi2_new](#) () const
Query for the chi2 value.
- void [print_to_file](#) (const std::string &basename) const
Function for printing the histogram, pcf, reference and normalization to file.
- void [print](#) () const
Print chi2 info to screen.
- void [weighted_analysis](#) (const [Library](#) &library, const std::vector< [BasisContainer](#) > &weights_table, const std::string &basename) const
Print weighted and unweighted data.

Private Member Functions

- void [read_reference](#) (const std::string &path)
Helper routine for reading the reference data from the input file.
- const double [get_r](#) (const int index) const
Helper routine for calculating the r value center of a bin index.

- const int [calculate_bin](#) (const double distance) const
Helper routine for calculating the bin for a given distance.
- const std::vector< double > [calculate_partial_histogram](#) (const [Library](#) &library, const int index) const
Helper routine for calculating the partial histogram for a basis element.
- const double [calculate_chi2](#) (const std::vector< double > &histogram) const
Helper routine for calculating the chi2 based on a histogram.

Private Attributes

- int [nsample_](#)
The number of elements in the sample set. Needed for normalization.
- double [rmin_](#)
The minimum r value.
- double [rmax_](#)
The maximum r value.
- double [dr_](#)
The r spacing (binwidth).
- double [one_over_dr_](#)
One over the r spacing (binwidth), for convenience.
- double [sigma_](#)
The sigma value to use for calculating chi2.
- double [one_over_sigma2_](#)
One over sigma squared, for convenience.
- double [numberdensity_](#)
The numberdensity for normalizing histograms to pcfs.
- std::pair< int, int > [partial_](#)
The partial to fit againts.
- double [chi2_](#)
The chi2 value.
- double [chi2_new_](#)
The chi2 value for the attempted move.

- `std::vector< double > pcf_reference_`
Holding the reference pcf from the data file.
- `std::vector< double > pcf_normalization_factor_`
Holding the factors to multiply each bin in the histograms with to get a pcf.
- `std::vector< double > histogram_`
Holding the current distance histogram.
- `std::vector< double > histogram_new_`
Holding the histogram for the attempted move.
- `std::pair< int, int > fit_interval_`
The interval (in bins) to use for calculating chi2.

Friends

- class [Test_PCFData](#)
Declaring the test class as friend, to facilitate testing.

3.12.1 Detailed Description

Class representing a PCF data set.

Definition at line 35 of file `pcfdata.h`.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 `PCFData::PCFData ()` `[inline]`

Default constructor for the [PCFData](#) class needed for initializing from the [Specswap](#) class.

Definition at line 42 of file `pcfdata.h`.

3.12.2.2 `PCFData::PCFData (const double rmin, const double rmax, const double dr, const double sigma, const double numberdensity, const std::pair< double, double > & fit_interval, const int nbins, const std::pair< int, int > & partial, const std::string & path)`

Constructor for the [PCFData](#) class.

Parameters

<i>rmin</i>	: The minimum r value.
<i>rmax</i>	: The maximum r value.
<i>dr</i>	: The bin size.
<i>sigma</i>	: The sigma value for calculating chi2.
<i>numberdensity</i>	: The numberdensity to use for the normalization.
<i>fit_interval</i>	: The interval to perform the fit within.
<i>nbins</i>	: The number of bins in the reference.
<i>partial</i>	: Indicating which partial in the library to fit.
<i>path</i>	: The full path to the file holding the reference.

Definition at line 33 of file pcfddata.cpp.

3.12.3 Member Function Documentation

3.12.3.1 void PCFData::accept ()

Function for indicating that the move should be accepted, which means chi2 should be set to chi2 new, and the data should be updated.

Definition at line 259 of file pcfddata.cpp.

3.12.3.2 const int PCFData::calculate_bin (const double *distance*) const [private]

Helper routine for calculating the bin for a given distance.

Parameters

<i>distance</i>	: The distance to calculate the bin for.
-----------------	--

Returns

: The bin.

Definition at line 126 of file pcfddata.cpp.

3.12.3.3 const double PCFData::calculate_chi2 (const std::vector< double > & *histogram*) const [private]

Helper routine for calculating the chi2 based on a histogram.

Parameters

<i>histogram</i>	: The histogram to calculate chi2 for.
------------------	--

Returns

: The calculated chi2.

Definition at line 181 of file pcfdata.cpp.

3.12.3.4 `const std::vector< double > PCFData::calculate_partial_histogram (const Library & library, const int index) const` [private]

Helper routine for calculating the partial histogram for a basis element.

Parameters

<i>library</i>	: The library to use.
----------------	-----------------------

Returns

: The calculated partial histogram.

Definition at line 134 of file pcfdata.cpp.

3.12.3.5 `double PCFData::get_chi2 () const` [inline]

Query for the chi2 value.

Returns

: chi2 .

Definition at line 89 of file pcfdata.h.

3.12.3.6 `double PCFData::get_chi2_new () const` [inline]

Query for the chi2 value.

Returns

: chi2 .

Definition at line 94 of file pcfdata.h.

3.12.3.7 `const double PCFData::get_r (const int index) const` [private]

Helper routine for calculating the r value center of a bin index.

Parameters

<i>index</i>	: The index to get the r value for.
--------------	-------------------------------------

Returns

: The center r value for a given bin index.

Definition at line 118 of file pcfddata.cpp.

3.12.3.8 void PCFData::init (const std::vector< int > & *sampleset*, const Library & *library*)

Function for setting up the pcf and calculating initial chi2.

Parameters

<i>sampleset</i>	: Vector holding the sample set indices.
<i>library</i>	: The library to take the data from.

Definition at line 206 of file pcfddata.cpp.

3.12.3.9 void PCFData::notify (const int *from_sample*, const int *from_basis*, const Library & *library*)

Function to notify the pcfddata object of an attempted move.

Parameters

<i>from_sample</i>	: The global index of the basis element to swap out of the sampleset.
<i>from_basis</i>	: The global index of the basis element to swap into the sampleset.
<i>library</i>	: The library to take the data from.

Definition at line 234 of file pcfddata.cpp.

3.12.3.10 void PCFData::print () const

Print chi2 info to screen.

Definition at line 275 of file pcfddata.cpp.

3.12.3.11 void PCFData::print_to_file (const std::string & *basename*) const

Function for printing the histogram, pcf, reference and normalization to file.

Parameters

<i>basename</i>	: The base name to save under.
-----------------	--------------------------------

Definition at line 286 of file pcfddata.cpp.

3.12.3.12 void PCFData::read_reference (const std::string & *path*) [private]

Helper routine for reading the reference data from the input file.

Parameters

<i>path</i>	: The full path to the file to read.
-------------	--------------------------------------

Definition at line 85 of file pcfdata.cpp.

3.12.3.13 void PCFData::weighted_analysis (const Library & *library*, const std::vector< BasisContainer > & *weights.table*, const std::string & *basename*) const

Print weighted and unweighted data.

Parameters

<i>library</i>	: The library to use.
<i>weights_ - table</i>	: The list of basis elements with weights and names.
<i>basename</i>	: The base file name to write to.

Definition at line 317 of file pcfdata.cpp.

3.12.4 Friends And Related Function Documentation

3.12.4.1 friend class Test_PCFData [friend]

Declaring the test class as friend, to facilitate testing.

Definition at line 200 of file pcfdata.h.

3.12.5 Member Data Documentation

3.12.5.1 double PCFData::chi2_ [private]

The chi2 value.

Definition at line 179 of file pcfdata.h.

3.12.5.2 double PCFData::chi2_new_ [private]

The chi2 value for the attempted move.

Definition at line 182 of file pcfdata.h.

3.12.5.3 double PCFData::dr_ [private]

The r spacing (binwidth).

Definition at line 161 of file pcfdata.h.

3.12.5.4 `std::pair<int,int> PCFData::fit_interval_` [private]

The interval (in bins) to use for calculating chi2.

Definition at line 197 of file pcfddata.h.

3.12.5.5 `std::vector<double> PCFData::histogram_` [private]

Holding the current distance histogram.

Definition at line 191 of file pcfddata.h.

3.12.5.6 `std::vector<double> PCFData::histogram_new_` [private]

Holding the histogram for the attempted move.

Definition at line 194 of file pcfddata.h.

3.12.5.7 `int PCFData::nsample_` [private]

The number of elements in the sample set. Needed for normalization.

Definition at line 152 of file pcfddata.h.

3.12.5.8 `double PCFData::numberdensity_` [private]

The numberdensity for normalizing histograms to pcfs.

Definition at line 173 of file pcfddata.h.

3.12.5.9 `double PCFData::one_over_dr_` [private]

One over the r spacing (binwidth), for convenience.

Definition at line 164 of file pcfddata.h.

3.12.5.10 `double PCFData::one_over_sigma2_` [private]

One over sigma squared, for convenience.

Definition at line 170 of file pcfddata.h.

3.12.5.11 `std::pair<int,int> PCFData::partial_` [private]

The partial to fit againts.

Definition at line 176 of file pcfddata.h.

3.12.5.12 `std::vector<double> PCFData::pcf_normalization_factor_` [private]

Holding the factors to multiply each bin in the histograms with to get a pcf.

Definition at line 188 of file pcfdata.h.

3.12.5.13 `std::vector<double> PCFData::pcf_reference_` [private]

Holding the reference pcf from the data file.

Definition at line 185 of file pcfdata.h.

3.12.5.14 `double PCFData::rmax_` [private]

The maximum r value.

Definition at line 158 of file pcfdata.h.

3.12.5.15 `double PCFData::rmin_` [private]

The minimum r value.

Definition at line 155 of file pcfdata.h.

3.12.5.16 `double PCFData::sigma_` [private]

The sigma value to use for calculating chi2.

Definition at line 167 of file pcfdata.h.

The documentation for this class was generated from the following files:

- [pcfdata.h](#)
- [pcfdata.cpp](#)

3.13 Sampleset Class Reference

Class representing the [Sampleset](#) in the SpecSwap simulation, for book keeping indices and performing swap moves.

```
#include <sampleset.h>
```

Public Member Functions

- [Sampleset](#) (const int nsample=0)
Default constructor needed for setup from specswap.

- void [add_index](#) (int index)
Used during setup to add elements to the sampleset.
- void [swap_into_slot](#) (int new_index, int slot)
Routine for performing the swap moves.
- const int [index_at](#) (const int slot)
Query function for the basis index at a specified slot.
- const std::vector< int > & [get_indices](#) () const
Query function for the index vector.
- const bool [is_added](#) (const int index) const
Function to check if an index is present in the sampleset.

Private Attributes

- std::vector< int > [indices_](#)
The indices vector, holding all indices in the sampleset.
- int [nsample_](#)
Total number of basis functions in the sampleset.

Friends

- class [Test_Sampleset](#)

3.13.1 Detailed Description

Class representing the [Sampleset](#) in the SpecSwap simulation, for book keeping indices and performing swap moves.

Definition at line 30 of file sampleset.h.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 [Sampleset::Sampleset](#) (const int *nsample* = 0)

Default constructor needed for setup from specswap.

Parameters

<i>nsample</i>	: The number of basis elements in the sample set.
----------------	---

Definition at line 32 of file sampleset.cpp.

3.13.3 Member Function Documentation

3.13.3.1 void Sampleset::add_index (int *index*)

Used during setup to add elements to the sampleset.

Parameters

<i>index</i>	: The corresponding index.
--------------	----------------------------

Definition at line 40 of file sampleset.cpp.

3.13.3.2 const std::vector<int>& Sampleset::get_indices () const [inline]

Query function for the index vector.

Returns

: The indices.

Definition at line 60 of file sampleset.h.

3.13.3.3 const int Sampleset::index_at (const int *slot*) [inline]

Query function for the basis index at a specified slot.

Parameters

<i>slot</i>	: The slot to the index at.
-------------	-----------------------------

Returns

: The corresponding index.

Definition at line 55 of file sampleset.h.

3.13.3.4 const bool Sampleset::is_added (const int *index*) const

Function to check if an index is present in the sampleset.

Returns

: True if the index is present in the sampleset, otherwise false.

Definition at line 62 of file sampleset.cpp.

3.13.3.5 void Sampleset::swap_into_slot (int *new_index*, int *slot*)

Routine for performing the swap moves.

Parameters

<i>new_index</i>	: The new index at the specified slot.
<i>slot</i>	: The slot in the sampleset to swap at.

Definition at line 53 of file sampleset.cpp.

3.13.4 Friends And Related Function Documentation

3.13.4.1 friend class Test_Sampleset [friend]

Definition at line 78 of file sampleset.h.

3.13.5 Member Data Documentation

3.13.5.1 std::vector<int> Sampleset::indices_ [private]

The indices vector, holding all indices in the sampleset.

Definition at line 72 of file sampleset.h.

3.13.5.2 int Sampleset::nsample_ [private]

Total number of basis functions in the sampleset.

Definition at line 75 of file sampleset.h.

The documentation for this class was generated from the following files:

- [sampleset.h](#)
- [sampleset.cpp](#)

3.14 ScalarDistributionData Class Reference

Class for representing a scalar distribution data set.

```
#include <scalardistributiondata.h>
```

Public Member Functions

- [ScalarDistributionData \(\)](#)

Default constructor needed for initialization from specswap.

- [ScalarDistributionData](#) (const std::string &name, const std::string &path, const double sigma, const int pos)
Constructor.
- void [init](#) (const std::vector< int > &sampleset, const [Library](#) &library)
Function for setting up the initial distribution and initial chi2.
- void [notify](#) (const int from_sample, const int from_basis, const [Library](#) &library)
Function to notify of an attempted move.
- void [accept](#) ()
Function for indicating that the move should be accepted, which means chi2 should be set to chi2 new, and the data should be updated.
- double [get_chi2](#) () const
Query for the chi2 value.
- double [get_chi2_new](#) () const
Query for the chi2 value.
- void [print](#) () const
Print chi2 info to screen.
- void [weighted_analysis](#) (const [Library](#) &library, const std::vector< [BasisContainer](#) > &weights_table, const std::string &basename) const
Print weighted and unweighted data.

Private Member Functions

- void [read_reference](#) (const std::string &path)
Helper routine to read in the reference.
- double [calculate_chi2](#) (const std::vector< double > &distribution) const
Helper routine to calculate chi2.
- int [get_bin](#) (const double value) const
Helper routine to calculate the bin for a given value.

Private Attributes

- std::string [name_](#)
The name (ending) of the scalar.

- `std::vector< double > target_`
The target distribution.
- `std::vector< double > scale_`
The scale (central bin values).
- `std::vector< double > factor_`
The factor for each point when calculating chi2.
- `std::vector< double > distribution_`
The distribution.
- `std::vector< double > distribution_new_`
The distribution for the attempted move.
- `double one_over_binsize_`
One over the binsize.
- `double lowest_`
Every thing lower goes in the first bin.
- `double highest_`
Every thing higher goes in the last bin.
- `double sigma_`
The sigma value.
- `double one_over_sigma2_`
The one over sigma2.
- `double chi2_`
The chi2 value.
- `double chi2_new_`
The chi2 value of the attempted move.
- `int pos_`
The index of the scalar in the library.
- `int nsample_`
The number of elements in the sample set.
- `int nbins_`
The number of bins in the distribution.

Friends

- class [Test_ScalarDistributionData](#)

Declaring the test class as friend to facilitate testing.

3.14.1 Detailed Description

Class for representing a scalar distribution data set.

Definition at line 34 of file `scalardistributiondata.h`.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 `ScalarDistributionData::ScalarDistributionData () [inline]`

Default constructor needed for initialization from `specswap`.

Definition at line 40 of file `scalardistributiondata.h`.

3.14.2.2 `ScalarDistributionData::ScalarDistributionData (const std::string & name, const std::string & path, const double sigma, const int pos)`

Constructor.

Parameters

<i>name</i>	: The name (ending) of the scalar.
<i>path</i>	: Path to the file holding the reference distribution.
<i>sigma</i>	: The sigma value.
<i>pos</i>	: The index of the scalar in the library.

Definition at line 34 of file `scalardistributiondata.cpp`.

3.14.3 Member Function Documentation

3.14.3.1 `void ScalarDistributionData::accept ()`

Function for indicating that the move should be accepted, which means `chi2` should be set to `chi2 new`, and the data should be updated.

Definition at line 196 of file `scalardistributiondata.cpp`.

3.14.3.2 `double ScalarDistributionData::calculate_chi2 (const std::vector< double > & distribution) const [private]`

Helper routine to calculate `chi2`.

Parameters

<i>value</i>	: The value to calculate for.
--------------	-------------------------------

Returns

: chi2

Definition at line 118 of file `scalardistributiondata.cpp`.

3.14.3.3 `int ScalarDistributionData::get_bin (const double value) const` `[private]`

Helper routine to calculate the bin for a given value.

Parameters

<i>value</i>	: The value to get the bin for.
--------------	---------------------------------

Returns

: The bin.

Definition at line 101 of file `scalardistributiondata.cpp`.

3.14.3.4 `double ScalarDistributionData::get_chi2 () const` `[inline]`

Query for the chi2 value.

Returns

: chi2 .

Definition at line 77 of file `scalardistributiondata.h`.

3.14.3.5 `double ScalarDistributionData::get_chi2_new () const` `[inline]`

Query for the chi2 value.

Returns

: chi2 .

Definition at line 82 of file `scalardistributiondata.h`.

3.14.3.6 `void ScalarDistributionData::init (const std::vector< int > & sampleset, const Library & library)`

Function for setting up the initial distribution and initial chi2.

Parameters

<i>sampleset</i>	: Vector holding the sample set indices.
<i>library</i>	: The library to take the data from.

Definition at line 141 of file `scalardistributiondata.cpp`.

3.14.3.7 void ScalarDistributionData::notify (const int *from_sample*, const int *from_basis*, const Library & *library*)

Function to notify of an attempted move.

Parameters

<i>from_samle</i>	: The global index of the basis element to swap out of the sampleset.
<i>from_basis</i>	: The global index of the basis element to swap into the sampleset.
<i>library</i>	: The library to take the data from.

Definition at line 171 of file `scalardistributiondata.cpp`.

3.14.3.8 void ScalarDistributionData::print () const

Print chi2 info to screen.

Definition at line 204 of file `scalardistributiondata.cpp`.

3.14.3.9 void ScalarDistributionData::read_reference (const std::string & *path*)
[private]

Helper routine to read in the reference.

Parameters

<i>path</i>	: The full path to the file holding the reference distribution.
-------------	---

Definition at line 50 of file `scalardistributiondata.cpp`.

3.14.3.10 void ScalarDistributionData::weighted_analysis (const Library & *library*, const std::vector< BasisContainer > & *weights_table*, const std::string & *basename*) const

Print weighted and unweighted data.

Parameters

<i>library</i>	: The library to use.
<i>weights_ - table</i>	: The list of basis elements with weights and names.
<i>basename</i>	: The base file name to write to.

Definition at line 214 of file `scalardistributiondata.cpp`.

3.14.4 Friends And Related Function Documentation

3.14.4.1 `friend class Test_ScalarDistributionData` `[friend]`

Declaring the test class as friend to facilitate testing.

Definition at line 168 of file `scalardistributiondata.h`.

3.14.5 Member Data Documentation

3.14.5.1 `double ScalarDistributionData::chi2_` `[private]`

The chi2 value.

Definition at line 153 of file `scalardistributiondata.h`.

3.14.5.2 `double ScalarDistributionData::chi2_new_` `[private]`

The chi2 value of the attempted move.

Definition at line 156 of file `scalardistributiondata.h`.

3.14.5.3 `std::vector<double> ScalarDistributionData::distribution_` `[private]`

The distribution.

Definition at line 132 of file `scalardistributiondata.h`.

3.14.5.4 `std::vector<double> ScalarDistributionData::distribution_new_` `[private]`

The distribution for the attempted move.

Definition at line 135 of file `scalardistributiondata.h`.

3.14.5.5 `std::vector<double> ScalarDistributionData::factor_` `[private]`

The factor for each point when calculating chi2.

Definition at line 129 of file `scalardistributiondata.h`.

3.14.5.6 `double ScalarDistributionData::highest_` `[private]`

Every thing higher goes in the last bin.

Definition at line 144 of file `scalardistributiondata.h`.

3.14.5.7 double ScalarDistributionData::lowest_ [private]

Every thing lower goes in the first bin.

Definition at line 141 of file `scalardistributiondata.h`.

3.14.5.8 std::string ScalarDistributionData::name_ [private]

The name (ending) of the scalar.

Definition at line 120 of file `scalardistributiondata.h`.

3.14.5.9 int ScalarDistributionData::nbins_ [private]

The number of bins in the distribution.

Definition at line 165 of file `scalardistributiondata.h`.

3.14.5.10 int ScalarDistributionData::nsample_ [private]

The number of elements in the sample set.

Definition at line 162 of file `scalardistributiondata.h`.

3.14.5.11 double ScalarDistributionData::one_over_binsize_ [private]

One over the binsize.

Definition at line 138 of file `scalardistributiondata.h`.

3.14.5.12 double ScalarDistributionData::one_over_sigma2_ [private]

The one over sigma2.

Definition at line 150 of file `scalardistributiondata.h`.

3.14.5.13 int ScalarDistributionData::pos_ [private]

The index of the scalar in the library.

Definition at line 159 of file `scalardistributiondata.h`.

3.14.5.14 std::vector<double> ScalarDistributionData::scale_ [private]

The scale (central bin values).

Definition at line 126 of file `scalardistributiondata.h`.

3.14.5.15 double ScalarDistributionData::sigma_ [private]

The sigma value.

Definition at line 147 of file `scalardistributiondata.h`.

3.14.5.16 std::vector<double> ScalarDistributionData::target_ [private]

The target distribution.

Definition at line 123 of file `scalardistributiondata.h`.

The documentation for this class was generated from the following files:

- [scalardistributiondata.h](#)
- [scalardistributiondata.cpp](#)

3.15 Specswap Class Reference

The central [Specswap](#) workhorse object performing moves and controlling communication with attached data sets.

```
#include <specswap.h>
```

Public Member Functions

- [Specswap](#) ()
Default constructor needed for setup from the rmc main program.
- [Specswap](#) (int nsample, const [Mlrnd](#) &rand, const std::string &library_path)
Normal constructor of the [Specswap](#) class.
- [Specswap](#) (int nsample, const [Mlrnd](#) &rand, const std::string &library_path, const std::string &restart_path)
Restart constructor of the [Specswap](#) class.
- void [add_scalar_mean](#) (const std::string &scalarname, const double target, const double sigma)
To add a mean scalar to fit.
- void [add_scalar_value](#) (const std::string &scalarname, const double value_low, const double value_high, const double fraction, const double sigma)
To add a scalar value to fit.
- void [add_scalar_distribution](#) (const std::string &scalarname, const std::string &filename, const double sigma)
To add a scalar distribution to fit.

- void [add_curve](#) (const double sigma, const bool area_renorm, const std::string &curve_name, const std::string &path)
Function for adding a curve data set.
- void [add_pcf](#) (const double rmin, const double rmax, const double dr, const double sigma, const double numberdensity, const std::pair< double, double > &fit_interval, const int nbins, const std::pair< int, int > &partial, const std::string &path)
Function for adding a PCF data set.
- void [setup](#) ()
The main setup function called when all input is read and the specswap object is to be prepared for running the simulation.
- void [move](#) ()
The move function called from the main RMC loop when a move should be made.
- const double [get_chi2](#) () const
Query for the chi2 value.
- const double [get_chi2_new](#) () const
Query for the new chi2 value.
- void [notify](#) ()
The notify function called from the main RMC loop when a move has been made and it is time to calculate the delta chi2.
- void [accept](#) ()
The accept function called from the main RMC loop when a move should be accepted and the new configuration stored as old.
- void [collect_weights](#) ()
Function for collecting weights by probing the sampleset.
- void [print](#) () const
The print function called from the main RMC loop to printout run info to standard out.
- void [print_start](#) () const
Printing startup information.
- void [print_stop](#) () const
Printing stop information.
- void [print_weights](#) (std::ofstream &output, const int weight) const
Print the weights to standard out.

- void [write_restart](#) (const std::string &path) const
Write restart info to the specified path.
- void [write_dump](#) (const std::string &title)
Write curve and pcf information for the present sampleset to file.
- void [write_weights_and_names_list](#) (const std::string &title)
Write a file with all basis names and their weights listed, sorted after highest weight.
- void [weighted_analysis](#) (const std::string &title)
Write all data sets to file, weighted and unweighted.
- void [chunk_analysis](#) (const std::string &title, const int chunks, const int chunk_size)
Write all data sets to file unweighted, summed in chunks of decreasing weights.

Private Member Functions

- void [setup_sampleset](#) ()
For setting up the sampleset.
- void [setup_chi2](#) ()
For setting up the chi2 values.
- void [print_to_file](#) (const std::string &basename) const
Function for printing run info to a file.
- const int [calc_scalar_pos](#) (const std::string &scalarnam) const
Obtain the position in the library for a given scalarnam.
- const int [random_basis](#) () const
Generates the index of a random basis from the library.
- double [random](#) () const
A random number generator.
- void [prepare_for_analysis](#) ()
Calling this function sets up things needed for all analysis.
- void [analyse_chunk](#) (const std::vector< [BasisContainer](#) > &weights_table, const std::string &basename)
Perform analysis on the chunk.

Private Attributes

- [MlRnd rand_](#)
The random number generator.
- [Library library_](#)
The library holding all basis element data.
- [Sampleset sampleset_](#)
The sampleset object.
- `std::vector< PCFData > pcf_data_`
The vector with objects holding PCF data.
- `std::vector< CurveData > curve_data_`
The vector with objects holding curve data.
- `std::vector< MeanScalarData > mean_scalar_data_`
The vector with mean scalar data objects.
- `std::vector< ValueScalarData > value_scalar_data_`
The vector with value scalar data objects.
- `std::vector< ScalarDistributionData > scalar_distribution_data_`
The vector with scalar distribution data objects.
- `int nsample_`
The number of basis elements in the sample set.
- `int ncurves_`
The number of curves in the library.
- `double chi2_`
The total specsrap chi2.
- `double chi2_new_`
The total new chi2 from specsrap.
- `std::vector< int > weights_`
The sampled weights multiplied with the number of times probed.
- `std::vector< BasisContainer > weights_table_`
Holding names with associated indices and weights for analysis.
- `int from_sample_`
Used to define a move, holding the index of the basis element to swap out of the sampleset.

- int [from_basis_](#)
Used to define a move, holding the index of the basis element to swap in to the sample set.
- int [slot_](#)
The sampleset slot to swap in.
- int [nprobe_](#)
The number of times weights have been collected.
- int [dump_counter_](#)
Counter for indexing filenames when dumping to file.
- bool [restart_](#)
Flag indicating if this is a restart run or not.
- std::string [restart_path_](#)
The path to the restart file to read from.

Friends

- class [Test_Specswap](#)
Declaring the test class as friend, to facilitate testing.

3.15.1 Detailed Description

The central [Specswap](#) workhorse object performing moves and controlling communication with attached data sets.

Definition at line 44 of file `specswap.h`.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 `Specswap::Specswap () [inline]`

Default constructor needed for setup from the rmc main program.

Definition at line 50 of file `specswap.h`.

3.15.2.2 `Specswap::Specswap (int nsample, const Mlnd & rand, const std::string & library_path)`

Normal constructor of the [Specswap](#) class.

Parameters

<i>nsample</i>	: The number of basis elements to take in the sample set.
<i>rand</i>	: The random number generator to use.
<i>library_path</i>	: The path to the library file.

Definition at line 34 of file specswap.cpp.

3.15.2.3 Specswap::Specswap (int *nsample*, const MlRnd & *rand*, const std::string & *library_path*, const std::string & *restart_path*)

Restart constructor of the [Specswap](#) class.

Parameters

<i>nsample</i>	: The number of basis elements to take in the sample set.
<i>rand</i>	: The random number generator to use.
<i>library_path</i>	: The path to the library file.
<i>restart_inpath</i>	: The path to the restart file.

Definition at line 58 of file specswap.cpp.

3.15.3 Member Function Documentation

3.15.3.1 void Specswap::accept ()

The accept function called from the main RMC loop when a move should be accepted and the new configuration stored as old.

Note

: Called from within the MC loop.

Definition at line 473 of file specswap.cpp.

3.15.3.2 void Specswap::add_curve (const double *sigma*, const bool *area_renorm*, const std::string & *curve_name*, const std::string & *path*)

Function for adding a curve data set.

Parameters

<i>sigma</i>	: The sigma value to use.
<i>area_renorm</i>	: Flag indicating if area renormalization should be used. <i>curve_name</i> : The name (ending) of the curve.
<i>path</i>	: Full path to the file holding the reference curve information.

Definition at line 157 of file specswap.cpp.

3.15.3.3 void Specswap::add_pcf (const double *rmin*, const double *rmax*, const double *dr*, const double *sigma*, const double *numberdensity*, const std::pair< double, double > & *fit_interval*, const int *nbins*, const std::pair< int, int > & *partial*, const std::string & *path*)

Function for adding a PCF data set.

Parameters

<i>rmin</i>	: The low r cutoff.
<i>rmax</i>	: The high r cutoff.
<i>dr</i>	: The r spacing.
<i>sigma</i>	: The sigma value to use for the dataset.
<i>numberdensity</i>	: The numberdensity to use for the normalization.
<i>fit_interval</i>	: The interval to use when calculating chi2. <i>nbins</i> : The number of bins in the data.
<i>partial</i>	: The partial in question, referring to the numbering in the library.
<i>path</i>	: Full path to the file holding the reference PCF information.

Definition at line 172 of file specswap.cpp.

3.15.3.4 void Specswap::add_scalar_distribution (const std::string & *scalarname*, const std::string & *filename*, const double *sigma*)

To add a scalar distribution to fit.

Parameters

<i>scalarname</i>	: The name of the scalar. Must match a scalar name in the added library.
<i>filename</i>	: The name of the file containing the reference data.
<i>sigma</i>	: The sigma value to use for the fit.

Definition at line 111 of file specswap.cpp.

3.15.3.5 void Specswap::add_scalar_mean (const std::string & *scalarname*, const double *target*, const double *sigma*)

To add a mean scalar to fit.

Parameters

<i>scalarname</i>	: The name of the scalar. Must match a scalar name in the added library.
<i>target</i>	: The target mean value to fit against.
<i>sigma</i>	: The sigma value to use for the fit.

Definition at line 86 of file specsrap.cpp.

3.15.3.6 void Specsrap::add_scalar_value (const std::string & *scalarname*, const double *value_low*, const double *value_high*, const double *fraction*, const double *sigma*)

To add a scalar value to fit.

Parameters

<i>scalarname</i>	: The name of the scalar. Must match a scalar name in the added library.
<i>value_low</i>	: The low limit of the fit interval.
<i>value_high</i>	: The high limit of the fit interval.
<i>fraction</i>	: The target fraction of elements with the specified scalar within the interval.
<i>sigma</i>	: The sigma value to use for the fit.

Definition at line 98 of file specsrap.cpp.

3.15.3.7 void Specsrap::analyse_chunk (const std::vector< BasisContainer > & *weights_table*, const std::string & *basename*) [private]

Perform analysis on the chunk.

Parameters

<i>weights_table</i>	: The chunk to analyse.
<i>basename</i>	: The base name of the file to write to.

Definition at line 903 of file specsrap.cpp.

3.15.3.8 const int Specsrap::calc_scalar_pos (const std::string & *scalarname*) const [private]

Obtain the position in the library for a given scalarname.

Parameters

<i>scalarname</i>	: The name of the scalar to check for.
-------------------	--

Returns

: The position of the scalar in the library.

Definition at line 122 of file specsrap.cpp.

3.15.3.9 void Specswap::chunk_analysis (const std::string & *title*, const int *chunks*, const int *chunk_size*)

Write all data sets to file unweighted, summed in chunks of decreasing weights.

Parameters

<i>title</i>	: The title of the run to use as part of the filename.
<i>chunks</i>	: The number of chunks to create.
<i>chunk_size</i>	: The size of the chunks.

Definition at line 835 of file specswap.cpp.

3.15.3.10 void Specswap::collect_weights ()

Function for collecting weights by probing the sample set.

Note

: Called from within the MC loop.

Definition at line 514 of file specswap.cpp.

3.15.3.11 const double Specswap::get_chi2 () const [inline]

Query for the chi2 value.

Note

: Called from within the MC loop.

Definition at line 162 of file specswap.h.

3.15.3.12 const double Specswap::get_chi2_new () const [inline]

Query for the new chi2 value.

Note

: Called from within the MC loop.

Definition at line 167 of file specswap.h.

3.15.3.13 void Specswap::move ()

The move function called from the main RMC loop when a move should be made.

Note

: Called from within the MC loop.

Definition at line 410 of file specsswap.cpp.

3.15.3.14 void Specsswap::notify ()

The notify function called from the main RMC loop when a move has been made and it is time to calculate the delta chi2.

Note

: Called from within the MC loop.

Definition at line 429 of file specsswap.cpp.

3.15.3.15 void Specsswap::prepare_for_analysis () [private]

Calling this function sets up things needed for all analysis.

Definition at line 752 of file specsswap.cpp.

3.15.3.16 void Specsswap::print () const

The print function called from the main RMC loop to printout run info to standard out.

Note

: Called from within the MC loop.

Definition at line 539 of file specsswap.cpp.

3.15.3.17 void Specsswap::print_start () const

Printing startup information.

Definition at line 621 of file specsswap.cpp.

3.15.3.18 void Specsswap::print_stop () const

Printing stop information.

Definition at line 630 of file specsswap.cpp.

3.15.3.19 void Specsswap::print_to_file (const std::string & *basename*) const [private]

Function for printing run info to a file.

Parameters

<i>basename</i>	: The path of the file to write to, withouth extension.
-----------------	---

Definition at line 639 of file specsswap.cpp.

3.15.3.20 void Specsswap::print_weights (std::ofstream & *output*, const int *weight*) const

Print the weights to standard out.

Parameters

<i>output</i>	: The ofstream to write to.
<i>weight</i>	: The weight to divide the incremented probe numbers with to produce the weights to print.

Definition at line 663 of file specsswap.cpp.

3.15.3.21 double Specsswap::random () const [private]

A random number generator.

Returns

: an integer between

Definition at line 736 of file specsswap.cpp.

3.15.3.22 const int Specsswap::random_basis () const [private]

Generates the index of a random basis from the library.

Returns

: A random basis index.

Definition at line 726 of file specsswap.cpp.

3.15.3.23 void Specsswap::setup ()

The main setup function called when all input is read and the specsswap object is to be prepared for running the simulation.

Definition at line 204 of file specsswap.cpp.

3.15.3.24 void Specswap::setup_chi2 () [private]

For setting up the chi2 values.

Definition at line 364 of file specswap.cpp.

3.15.3.25 void Specswap::setup_sampleset () [private]

For setting up the sampleset.

Definition at line 245 of file specswap.cpp.

3.15.3.26 void Specswap::weighted_analysis (const std::string & title)

Write all data sets to file, weighted and unweighted.

Parameters

<i>title</i>	: The title of the run to use as part of the filename.
--------------	--

Definition at line 817 of file specswap.cpp.

3.15.3.27 void Specswap::write_dump (const std::string & title)

Write curve and pcf information for the present sampleset to file.

Parameters

<i>title</i>	: The title of the run to use as part of the filename.
--------------	--

Definition at line 607 of file specswap.cpp.

3.15.3.28 void Specswap::write_restart (const std::string & path) const

Write restart info to the specified path.

Definition at line 688 of file specswap.cpp.

3.15.3.29 void Specswap::write_weights_and_names_list (const std::string & title)

Write a file with all basis names and their weights listed, sorted after highest weight.

Parameters

<i>title</i>	: The title of the run to use as part of the filename.
--------------	--

Definition at line 781 of file specswap.cpp.

3.15.4 Friends And Related Function Documentation

3.15.4.1 friend class Test_Specswap [friend]

Declaring the test class as friend, to facilitate testing.

Definition at line 368 of file specswap.h.

3.15.5 Member Data Documentation

3.15.5.1 double Specswap::chi2_ [private]

The total specswap chi2.

Definition at line 335 of file specswap.h.

3.15.5.2 double Specswap::chi2_new_ [private]

The total new chi2 from specswap.

Definition at line 338 of file specswap.h.

3.15.5.3 std::vector<CurveData> Specswap::curve_data_ [private]

The vector with objects holding curve data.

Definition at line 317 of file specswap.h.

3.15.5.4 int Specswap::dump_counter_ [private]

Counter for indexing filenames when dumping to file.

Definition at line 359 of file specswap.h.

3.15.5.5 int Specswap::from_basis_ [private]

Used to define a move, holding the index of the basis element to swap in to the sample set.

Definition at line 350 of file specswap.h.

3.15.5.6 int Specswap::from_sample_ [private]

Used to define a move, holding the index of the basis element to swap out of the sample set.

Definition at line 347 of file specswap.h.

3.15.5.7 Library Specsrap::library_ [private]

The library holding all basis element data.

Definition at line 308 of file specsrap.h.

3.15.5.8 std::vector<MeanScalarData> Specsrap::mean_scalar_data_ [private]

The vector with mean scalar data objects.

Definition at line 320 of file specsrap.h.

3.15.5.9 int Specsrap::ncurves_ [private]

The number of curves in the library.

Definition at line 332 of file specsrap.h.

3.15.5.10 int Specsrap::nprobe_ [private]

The number of times weights have been collected.

Definition at line 356 of file specsrap.h.

3.15.5.11 int Specsrap::nsample_ [private]

The number of basis elements in the sample set.

Definition at line 329 of file specsrap.h.

3.15.5.12 std::vector<PCFData> Specsrap::pcf_data_ [private]

The vector with objects holding PCF data.

Definition at line 314 of file specsrap.h.

3.15.5.13 Mlrand Specsrap::rand_ [private]

The random number generator.

Definition at line 305 of file specsrap.h.

3.15.5.14 bool Specsrap::restart_ [private]

Flag indicating if this is a restart run or not.

Definition at line 362 of file specsrap.h.

3.15.5.15 `std::string Specswap::restart_path_` [private]

The path to the restart file to read from.

Definition at line 365 of file `specswap.h`.

3.15.5.16 `Sampleset Specswap::sampleset_` [private]

The sampleset object.

Definition at line 311 of file `specswap.h`.

3.15.5.17 `std::vector<ScalarDistributionData> Specswap::scalar_distribution_data_` [private]

The vector with scalar distribution data objects.

Definition at line 326 of file `specswap.h`.

3.15.5.18 `int Specswap::slot_` [private]

The sampleset slot to swap in.

Definition at line 353 of file `specswap.h`.

3.15.5.19 `std::vector<ValueScalarData> Specswap::value_scalar_data_` [private]

The vector with value scalar data objects.

Definition at line 323 of file `specswap.h`.

3.15.5.20 `std::vector<int> Specswap::weights_` [private]

The sampled weights multiplied with the number of times probed.

Definition at line 341 of file `specswap.h`.

3.15.5.21 `std::vector<BasisContainer> Specswap::weights_table_` [private]

Holding names with associated indices and weights for analysis.

Definition at line 344 of file `specswap.h`.

The documentation for this class was generated from the following files:

- [specswap.h](#)
- [specswap.cpp](#)

3.16 ValueScalarData Class Reference

Class representing a value scalar data set.

```
#include <valuescaldat.h>
```

Public Member Functions

- [ValueScalarData](#) ()
Default constructor needed for initialization from specswap.
- [ValueScalarData](#) (const std::string &name, const std::pair< double, double > &interval, const double target, const double sigma, const int pos)
Constructor.
- void [init](#) (const std::vector< int > &sampleset, const [Library](#) &library)
Function for setting up the initial mean value and initial chi2.
- void [notify](#) (const int from_sample, const int from_basis, const [Library](#) &library)
Function to notify of an attempted move.
- void [accept](#) ()
Function for indicating that the move should be accepted, which means chi2 should be set to chi2 new, and the data should be updated.
- double [get_chi2](#) () const
Query for the chi2 value.
- double [get_chi2_new](#) () const
Query for the chi2 value.
- void [print](#) () const
Print chi2 info to screen.
- void [weighted_analysis](#) (const [Library](#) &library, const std::vector< [BasisContainer](#) > &weights_table, const std::string &basename) const
Print weighted and unweighted data.

Private Member Functions

- double [calculate_chi2](#) (const double value) const
Helper routine to calculate chi2.

Private Attributes

- `std::string name_`
The name (ending) of the scalar.
- `std::pair< double, double > interval_`
The interval we are fitting to.
- `double target_`
The target value.
- `double sigma_`
The sigma value.
- `double one_over_sigma2_`
The one over sigma2.
- `double value_`
The value.
- `double value_new_`
The value for the attempted move.
- `double chi2_`
The chi2 value.
- `double chi2_new_`
The chi2 value of the attempted move.
- `int pos_`
The index of the scalar in the library.
- `int nsample_`
The number of elements in the sample set.

Friends

- class `Test_ValueScalarData`
Declaring the test class as friend to facilitate testing.

3.16.1 Detailed Description

Class representing a value scalar data set.

Definition at line 34 of file `valuescalardata.h`.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 ValueScalarData::ValueScalarData () [inline]

Default constructor needed for initialization from specswap.

Definition at line 40 of file valuescalardata.h.

3.16.2.2 ValueScalarData::ValueScalarData (const std::string & *name*, const std::pair< double, double > & *interval*, const double *target*, const double *sigma*, const int *pos*)

Constructor.

Parameters

<i>name</i>	: The name (ending) of the scalar.
<i>interval</i>	: The bounds the fit concern.
<i>target</i>	: Target fraction value.
<i>sigma</i>	: The sigma value.
<i>pos</i>	: The index of the scalar in the library.

Definition at line 33 of file valuescalardata.cpp.

3.16.3 Member Function Documentation

3.16.3.1 void ValueScalarData::accept ()

Function for indicating that the move should be accepted, which means chi2 should be set to chi2 new, and the data should be updated.

Definition at line 117 of file valuescalardata.cpp.

3.16.3.2 double ValueScalarData::calculate_chi2 (const double *value*) const [private]

Helper routine to calculate chi2.

Parameters

<i>value</i>	: The value to calculate for.
--------------	-------------------------------

Returns

: chi2

Definition at line 50 of file valuescalardata.cpp.

3.16.3.3 double ValueScalarData::get_chi2 () const [inline]

Query for the chi2 value.

Returns

: chi2 .

Definition at line 79 of file valuescalardata.h.

3.16.3.4 double ValueScalarData::get_chi2_new () const [inline]

Query for the chi2 value.

Returns

: chi2 .

Definition at line 84 of file valuescalardata.h.

3.16.3.5 void ValueScalarData::init (const std::vector< int > & sampleset, const Library & library)

Function for setting up the initial mean value and initial chi2.

Parameters

<i>sampleset</i>	: Vector holding the sample set indices.
<i>library</i>	: The library to take the data from.

Definition at line 58 of file valuescalardata.cpp.

3.16.3.6 void ValueScalarData::notify (const int from_sample, const int from_basis, const Library & library)

Function to notify of an attempted move.

Parameters

<i>from_samle</i>	: The global index of the basis element to swap out of the sampleset.
<i>from_basis</i>	: The global index of the basis element to swap into the sampleset.
<i>library</i>	: The library to take the data from.

Definition at line 92 of file valuescalardata.cpp.

3.16.3.7 void ValueScalarData::print () const

Print chi2 info to screen.

Definition at line 125 of file valuescalardata.cpp.

3.16.3.8 `void ValueScalarData::weighted_analysis (const Library & library, const std::vector< BasisContainer > & weights_table, const std::string & basename) const`

Print weighted and unweighted data.

Parameters

<i>library</i>	: The library to use.
<i>weights_table</i>	: The list of basis elements with weights and names.
<i>basename</i>	: The base file name to write to.

Definition at line 144 of file valuescalardata.cpp.

3.16.4 Friends And Related Function Documentation

3.16.4.1 `friend class Test_ValueScalarData [friend]`

Declaring the test class as friend to facilitate testing.

Definition at line 133 of file valuescalardata.h.

3.16.5 Member Data Documentation

3.16.5.1 `double ValueScalarData::chi2_ [private]`

The chi2 value.

Definition at line 124 of file valuescalardata.h.

3.16.5.2 `double ValueScalarData::chi2_new_ [private]`

The chi2 value of the attempted move.

Definition at line 126 of file valuescalardata.h.

3.16.5.3 `std::pair<double,double> ValueScalarData::interval_ [private]`

The interval we are fitting to.

Definition at line 112 of file valuescalardata.h.

3.16.5.4 `std::string ValueScalarData::name_ [private]`

The name (ending) of the scalar.

Definition at line 110 of file valuescalardata.h.

3.16.5.5 int ValueScalarData::nsample_ [private]

The number of elements in the sample set.

Definition at line 130 of file valuescalardata.h.

3.16.5.6 double ValueScalarData::one_over_sigma2_ [private]

The one over sigma2.

Definition at line 118 of file valuescalardata.h.

3.16.5.7 int ValueScalarData::pos_ [private]

The index of the scalar in the library.

Definition at line 128 of file valuescalardata.h.

3.16.5.8 double ValueScalarData::sigma_ [private]

The sigma value.

Definition at line 116 of file valuescalardata.h.

3.16.5.9 double ValueScalarData::target_ [private]

The target value.

Definition at line 114 of file valuescalardata.h.

3.16.5.10 double ValueScalarData::value_ [private]

The value.

Definition at line 120 of file valuescalardata.h.

3.16.5.11 double ValueScalarData::value_new_ [private]

The value for the attempted move.

Definition at line 122 of file valuescalardata.h.

The documentation for this class was generated from the following files:

- [valuescalardata.h](#)
- [valuescalardata.cpp](#)

Chapter 4

File Documentation

4.1 basiscontainer.h File Reference

File for the [BasisContainer](#) definition and implementation.

Classes

- struct [BasisContainer](#)

A minimal struct to bunch together an index, a weight and a basis name.

4.1.1 Detailed Description

File for the [BasisContainer](#) definition and implementation.

Definition in file [basiscontainer.h](#).

4.2 curvedata.cpp File Reference

File for the [CurveData](#) class implementation.

```
#include <fstream>
#include <cmath>
#include "ioutils.h"
#include "mathutils.h"
#include "library.h"
#include "curvedata.h"
```

4.2.1 Detailed Description

File for the [CurveData](#) class implementation.

Definition in file [curvedata.cpp](#).

4.3 curvedata.h File Reference

File for the [CurveData](#) class definition.

```
#include <vector>
#include <string>
#include "basiscontainer.h"
```

Classes

- class [CurveData](#)
Class representing a curve data set.

4.3.1 Detailed Description

File for the [CurveData](#) class definition.

Definition in file [curvedata.h](#).

4.4 ioexception.cpp File Reference

File for the [IOException](#) class implementation.

```
#include <cstdio>
#include "ioexception.h"
#include "ioutils.h"
```

4.4.1 Detailed Description

File for the [IOException](#) class implementation.

Definition in file [ioexception.cpp](#).

4.5 ioexception.h File Reference

File for the [IOException](#) class definition.


```
#include <string>
#include <stdexcept>
```

Classes

- class [IOException](#)
Class for representing an IO exception.

4.5.1 Detailed Description

File for the [IOException](#) class definition.

Definition in file [ioexception.h](#).

4.6 ioutils.cpp File Reference

File containing implementations for functions defined in [ioutils.h](#).

```
#include <cstdlib>
#include <cstdio>
#include <ctime>
#include <stdexcept>
#include <sstream>
#include "ioexception.h"
#include "ioutils.h"
```

Functions

- std::string [to_string](#) (const int i)
Convert an integer to string.
- std::string [newline_indent](#) (const std::string &message, const int blanks)
Insert a number of blank spaces after each newline.
- void [print_startup](#) ()
Prints the startup message to stdout.
- bool [eof](#) (std::ifstream &stream)
Checks if the stream is at the end.
- void [check_path](#) (const std::string &path, const std::string &location)

Checks that a file exists and can be opened for reading.

- void `check_eof` (std::ifstream &stream, const std::string &filename, const std::string &location)

Checks if the stream is at the end and rais an error if it is.

- void `check_sigma` (const double sigma, const std::string &location)

Checks that a given sigma value is not too close to zero.

- void `check_positive_integer` (const int value, const std::string &filename, const std::string &location)

Checks that a given integer value is above or equal to zero.

- void `open_file_error` (const std::string &filename, const std::string &location)

Raise an error for problems with opening a file.

- void `empty_file_error` (const std::string &filename, const std::string &location)

Raise an error for an empty file.

- void `unknown_section_error` (const std::string §ion, const std::string &location)

Raise an error for an unknown section.

- void `same_section_error` (const std::string §ion, const std::string &location)

Raise an error for encountering the same section twice.

- void `missing_keyword_error` (const std::string &keyword, const std::string &location)

Raise an error for missing keyword.

- void `same_keyword_error` (const std::string &keyword, const std::string &location)

Raise an error for finding the same keyword twice.

- void `unknown_keyword_error` (const std::string &keyword, const std::string &location)

Raise an error for unknown keyword.

- void `read_keyword_error` (const std::string &filename, const std::string &expected, const std::string &found, const std::string &location)

Raise an error for problems with reading a keyword.

- void `start_timer` ()

Sets the global time0. (Starts the timer.)

- int `timer` ()

Function used for timing.

- void [timestamp](#) ()
Prints wall clock time and date info to stdout.
- void [error](#) (const std::string &message, const std::string &location)
Raises an error.
- void [error_exit](#) (const std::string &program)
Function for exiting the program with specified exit code and a short error message.
- void [eatline](#) (std::ifstream &infile)
Function for eating (skipping) a line in an ifstream.
- void [print_success](#) ()
Prints the exit message to stdout.

Variables

- int [time_0__](#)

4.6.1 Detailed Description

File containing implementations for functions defined in [ioutils.h](#).

Definition in file [ioutils.cpp](#).

4.6.2 Function Documentation

4.6.2.1 void [check_eof](#) (std::ifstream & *stream*, const std::string & *filename*, const std::string & *location* = " ")

Checks if the stream is at the end and rais an error if it is.

Parameters

<i>stream</i>	: The stream to check.
<i>filename</i>	: The associated filename.

Definition at line 161 of file [ioutils.cpp](#).

4.6.2.2 void [check_path](#) (const std::string & *path*, const std::string & *location* = " ")

Checks that a file exists and can be opened for reading.

Parameters

<i>path</i>	: The path to check.
<i>location</i>	: The location of the check.

Definition at line 137 of file ioutils.cpp.

4.6.2.3 void check_positive_integer (const int *value*, const std::string & *filename*, const std::string & *location* = " ")

Checks that a given integer value is above or equal to zero.

Parameters

<i>value</i>	: The value to check.
<i>filename</i>	: The name of the file that was read.
<i>location</i>	: The location of the check.

Definition at line 190 of file ioutils.cpp.

4.6.2.4 void check_sigma (const double *sigma*, const std::string & *location* = " ")

Checks that a given sigma value is not too close to zero.

Parameters

<i>sigma</i>	: The value to check.
<i>location</i>	: The location of the check.

Definition at line 173 of file ioutils.cpp.

4.6.2.5 void eatline (std::ifstream & *infile*)

Function for eating (skipping) a line in an ifstream.

Definition at line 328 of file ioutils.cpp.

4.6.2.6 void empty_file_error (const std::string & *filename*, const std::string & *location* = " ")

Raise an error for an empty file.

Parameters

<i>filename</i>	: The file name.
<i>location</i>	: The location of the error.

Definition at line 211 of file ioutils.cpp.

4.6.2.7 bool eof (std::ifstream & *stream*)

Checks if the stream is at the end.

Parameters

<i>stream</i>	: The stream to check.
---------------	------------------------

Returns

: If it is at end of file or not.

Definition at line 125 of file ioutils.cpp.

4.6.2.8 void error (const std::string & *message*, const std::string & *location* = std::string("No location provided."))

Raises an error.

Parameters

<i>message</i>	: The error message.
<i>location</i>	: The location of the error.

Definition at line 311 of file ioutils.cpp.

4.6.2.9 void error_exit (const std::string & *program* = "SpecSwap-RMC")

Function for exiting the program with specified exit code and a short error message.

Parameters

<i>program</i>	: The name of the program to exit.
----------------	------------------------------------

Definition at line 319 of file ioutils.cpp.

4.6.2.10 void missing_keyword_error (const std::string & *keyword*, const std::string & *location* = " ")

Raise an error for missing keyword.

Parameters

<i>message</i>	: The missing keyword.
<i>location</i>	: The location of the error.

Definition at line 238 of file ioutils.cpp.

4.6.2.11 `std::string newline_indent (const std::string & message, const int blanks)`

Insert a number of blank spaces after each newline.

Parameters

<i>message</i>	: The string to add blanks to.
<i>blanks</i>	: The number of blanks to add after each newline.

Returns

: The processed string.

Definition at line 46 of file ioutils.cpp.

4.6.2.12 `void open_file_error (const std::string & filename, const std::string & location = " ")`

Raise an error for problems with opening a file.

Parameters

<i>filename</i>	: The file name.
<i>location</i>	: The location of the error.

Definition at line 202 of file ioutils.cpp.

4.6.2.13 `void print_startup ()`

Prints the startup message to stdout.

Definition at line 77 of file ioutils.cpp.

4.6.2.14 `void print_success ()`

Prints the exit message to stdout.

Definition at line 340 of file ioutils.cpp.

4.6.2.15 `void read_keyword_error (const std::string & filename, const std::string & expected, const std::string & found, const std::string & location = " ")`

Raise an error for problems with reading a keyword.

Parameters

<i>filename</i>	: The file that was read from.
<i>expected</i>	: The expected keyword.
<i>found</i>	: The the encountered keyword.
<i>location</i>	: The location of the error.

Definition at line 265 of file ioutils.cpp.

4.6.2.16 `void same_keyword_error (const std::string & keyword, const std::string & location = " ")`

Raise an error for finding the same keyword twice.

Parameters

<i>message</i>	: The keyword that was found twice.
<i>location</i>	: The location of the error.

Definition at line 247 of file ioutils.cpp.

4.6.2.17 `void same_section_error (const std::string & section, const std::string & location = " ")`

Raise an error for encountering the same section twice.

Parameters

<i>message</i>	: The section keyword.
<i>location</i>	: The location of the error.

Definition at line 229 of file ioutils.cpp.

4.6.2.18 `void start_timer ()`

Sets the global time0. (Starts the timer.)

Definition at line 280 of file ioutils.cpp.

4.6.2.19 `int timer ()`

Function used for timing.

Returns

: Seconds since the timer was started.

Definition at line 289 of file ioutils.cpp.

4.6.2.20 `void timestamp ()`

Prints wall clock time and date info to stdout.

Definition at line 298 of file ioutils.cpp.

4.6.2.21 std::string to_string (const int i)

Convert an integer to string.

Parameters

<i>i</i>	: The integer to convert.
----------	---------------------------

Returns

: A string representation of the integer.

Definition at line 35 of file ioutils.cpp.

4.6.2.22 void unknown_keyword_error (const std::string & keyword, const std::string & location = " ")

Raise an error for unknown keyword.

Parameters

<i>message</i>	: The keyword that was found.
<i>location</i>	: The location of the error.

Definition at line 256 of file ioutils.cpp.

4.6.2.23 void unknown_section_error (const std::string & section, const std::string & location = " ")

Raise an error for an unknown section.

Parameters

<i>message</i>	: The unknown section keyword.
<i>location</i>	: The location of the error.

Definition at line 220 of file ioutils.cpp.

4.6.3 Variable Documentation**4.6.3.1 int time_0__**

Definition at line 31 of file ioutils.cpp.

4.7 ioutils.h File Reference

File containing definitions of macros and utility functions for IO and error handling.


```
#include <fstream>
#include <string>
```

Defines

- #define [LOCATION](#) std::string(std::string(__FILE__) + ": " + to_string(__LINE__ -
))
Defines a macro for giving an error with specified location.
- #define [FUNCTION](#) std::string(__func__)
Defines a macro for giving an error with specified function.

Functions

- std::string [to_string](#) (const int i)
Convert an integer to string.
- std::string [newline_indent](#) (const std::string &message, const int blanks)
Insert a number of blank spaces after each newline.
- bool [eof](#) (std::ifstream &stream)
Checks if the stream is at the end.
- void [check_eof](#) (std::ifstream &stream, const std::string &filename, const std::string
&location="")
Checks if the stream is at the end and rais an error if it is.
- void [check_path](#) (const std::string &path, const std::string &location="")
Checks that a file exists and can be opened for reading.
- void [check_sigma](#) (const double sigma, const std::string &location="")
Checks that a given sigma value is not too close to zero.
- void [check_positive_integer](#) (const int value, const std::string &filename, const
std::string &location="")
Checks that a given integer value is above or equal to zero.
- void [open_file_error](#) (const std::string &filename, const std::string &location="")
Raise an error for problems with opening a file.
- void [empty_file_error](#) (const std::string &filename, const std::string &location="")
Raise an error for an empty file.

- void `unknown_section_error` (const std::string §ion, const std::string &location="")
Raise an error for an unknown section.
- void `same_section_error` (const std::string §ion, const std::string &location="")
Raise an error for encountering the same section twice.
- void `missing_keyword_error` (const std::string &keyword, const std::string &location="")
Raise an error for missing keyword.
- void `same_keyword_error` (const std::string &keyword, const std::string &location="")
Raise an error for finding the same keyword twice.
- void `unknown_keyword_error` (const std::string &keyword, const std::string &location="")
Raise an error for unknown keyword.
- void `read_keyword_error` (const std::string &filename, const std::string &expected, const std::string &found, const std::string &location="")
Raise an error for problems with reading a keyword.
- void `start_timer` ()
Sets the global time0. (Starts the timer.)
- int `timer` ()
Function used for timing.
- void `timestamp` ()
Prints wall clock time and date info to stdout.
- void `error` (const std::string &message, const std::string &location=std::string("No location provided."))
Raises an error.
- void `error_exit` (const std::string &program="SpecSwap-RMC")
Function for exiting the program with specified exit code and a short error message.
- void `print_startup` ()
Prints the startup message to stdout.
- void `eatline` (std::ifstream &infile)
Function for eating (skipping) a line in an ifstream.
- void `print_success` ()

Prints the exit message to stdout.

4.7.1 Detailed Description

File containing definitions of macros and utility functions for IO and error handling.

Definition in file [ioutils.h](#).

4.7.2 Define Documentation

4.7.2.1 `#define FUNCTION std::string(...func...)`

Defines a macro for giving an error with specified function.

Definition at line 32 of file ioutils.h.

4.7.2.2 `#define LOCATION std::string(std::string(__FILE__) + ": " + to_string(__LINE__))`

Defines a macro for giving an error with specified location.

Definition at line 29 of file ioutils.h.

4.7.3 Function Documentation

4.7.3.1 `void check_eof (std::ifstream & stream, const std::string & filename, const std::string & location = " ")`

Checks if the stream is at the end and rais an error if it is.

Parameters

<i>stream</i>	: The stream to check.
<i>filename</i>	: The associated filename.

Definition at line 161 of file ioutils.cpp.

4.7.3.2 `void check_path (const std::string & path, const std::string & location = " ")`

Checks that a file exists and can be opened for reading.

Parameters

<i>path</i>	: The path to check.
<i>location</i>	: The location of the check.

Definition at line 137 of file ioutils.cpp.

4.7.3.3 void check_positive_integer (const int *value*, const std::string & *filename*, const std::string & *location* = " ")

Checks that a given integer value is above or equal to zero.

Parameters

<i>value</i>	: The value to check.
<i>filename</i>	: The name of the file that was read.
<i>location</i>	: The location of the check.

Definition at line 190 of file ioutils.cpp.

4.7.3.4 void check_sigma (const double *sigma*, const std::string & *location* = " ")

Checks that a given sigma value is not too close to zero.

Parameters

<i>sigma</i>	: The value to check.
<i>location</i>	: The location of the check.

Definition at line 173 of file ioutils.cpp.

4.7.3.5 void eatline (std::ifstream & *infile*)

Function for eating (skipping) a line in an ifstream.

Definition at line 328 of file ioutils.cpp.

4.7.3.6 void empty_file_error (const std::string & *filename*, const std::string & *location* = " ")

Raise an error for an empty file.

Parameters

<i>filename</i>	: The file name.
<i>location</i>	: The location of the error.

Definition at line 211 of file ioutils.cpp.

4.7.3.7 bool eof (std::ifstream & *stream*)

Checks if the stream is at the end.

Parameters

<i>stream</i>	: The stream to check.
---------------	------------------------

Returns

: If it is at end of file or not.

Definition at line 125 of file ioutils.cpp.

4.7.3.8 `void error (const std::string & message, const std::string & location = std::string("No location provided."))`

Raises an error.

Parameters

<i>message</i>	: The error message.
<i>location</i>	: The location of the error.

Definition at line 311 of file ioutils.cpp.

4.7.3.9 `void error_exit (const std::string & program = "SpecSwap-RMC")`

Function for exiting the program with specified exit code and a short error message.

Parameters

<i>program</i>	: The name of the program to exit.
----------------	------------------------------------

Definition at line 319 of file ioutils.cpp.

4.7.3.10 `void missing_keyword_error (const std::string & keyword, const std::string & location = "")`

Raise an error for missing keyword.

Parameters

<i>message</i>	: The missing keyword.
<i>location</i>	: The location of the error.

Definition at line 238 of file ioutils.cpp.

4.7.3.11 `std::string newline_indent (const std::string & message, const int blanks)`

Insert a number of blank spaces after each newline.

Parameters

<i>message</i>	: The string to add blanks to.
<i>blanks</i>	: The number of blanks to add after each newline.

Returns

: The processed string.

Definition at line 46 of file ioutils.cpp.

4.7.3.12 void open_file_error (const std::string & filename, const std::string & location = " ")

Raise an error for problems with opening a file.

Parameters

<i>filename</i>	: The file name.
<i>location</i>	: The location of the error.

Definition at line 202 of file ioutils.cpp.

4.7.3.13 void print_startup ()

Prints the startup message to stdout.

Definition at line 77 of file ioutils.cpp.

4.7.3.14 void print_success ()

Prints the exit message to stdout.

Definition at line 340 of file ioutils.cpp.

4.7.3.15 void read_keyword_error (const std::string & filename, const std::string & expected, const std::string & found, const std::string & location = " ")

Raise an error for problems with reading a keyword.

Parameters

<i>filename</i>	: The file that was read from.
<i>expected</i>	: The expected keyword.
<i>found</i>	: The the encountered keyword.
<i>location</i>	: The location of the error.

Definition at line 265 of file ioutils.cpp.

4.7.3.16 void same_keyword_error (const std::string & keyword, const std::string & location = " ")

Raise an error for finding the same keyword twice.

Parameters

<i>message</i>	: The keyword that was found twice.
<i>location</i>	: The location of the error.

Definition at line 247 of file ioutils.cpp.

4.7.3.17 `void same_section_error (const std::string & section, const std::string & location = " ")`

Raise an error for encountering the same section twice.

Parameters

<i>message</i>	: The section keyword.
<i>location</i>	: The location of the error.

Definition at line 229 of file ioutils.cpp.

4.7.3.18 `void start_timer ()`

Sets the global time0. (Starts the timer.)

Definition at line 280 of file ioutils.cpp.

4.7.3.19 `int timer ()`

Function used for timing.

Returns

: Seconds since the timer was started.

Definition at line 289 of file ioutils.cpp.

4.7.3.20 `void timestamp ()`

Prints wall clock time and date info to stdout.

Definition at line 298 of file ioutils.cpp.

4.7.3.21 `std::string to_string (const int i)`

Convert an integer to string.

Parameters

<i>i</i>	: The integer to convert.
----------	---------------------------

Returns

: A string representation of the integer.

Definition at line 35 of file ioutils.cpp.

4.7.3.22 void unknown_keyword_error (const std::string & *keyword*, const std::string & *location* = " ")

Raise an error for unknown keyword.

Parameters

<i>message</i>	: The keyword that was found.
<i>location</i>	: The location of the error.

Definition at line 256 of file ioutils.cpp.

4.7.3.23 void unknown_section_error (const std::string & *section*, const std::string & *location* = " ")

Raise an error for an unknown section.

Parameters

<i>message</i>	: The unknown section keyword.
<i>location</i>	: The location of the error.

Definition at line 220 of file ioutils.cpp.

4.8 library.cpp File Reference

File for the [Library](#) class definition.

```
#include <algorithm>
#include <fstream>
#include <iostream>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include "ioutils.h"
#include "mathutils.h"
#include "library.h"
```


4.8.1 Detailed Description

File for the [Library](#) class definition.

Definition in file [library.cpp](#).

4.9 library.h File Reference

File for the [Library](#) class definition.

```
#include <string>
#include <vector>
#include "matrix.h"
```

Classes

- struct [Convolute](#)
Struct for encapsulating the convolute parameters.
- struct [Format](#)
Struct for encapsulating the format parameters.
- class [Library](#)
The class defining the library to use in SpecSwap-RMC.

4.9.1 Detailed Description

File for the [Library](#) class definition.

Definition in file [library.h](#).

4.10 mathutils.cpp File Reference

File containing implementations of the utility functions defined in [mathutils.h](#).

```
#include <cmath>
#include <cstdio>
#include "matrix.h"
#include "mathutils.h"
```

Functions

- double [calculate_distance](#) (const double point1[3], const double point2[3])
Function to calculate the distance between two points.
- [Matrix setup_index_matrix](#) (const int ntypes)
Generate the index matrix that maps partials to array indiceds.
- double [vsum](#) (std::vector< double > const &data)
Sum the elements of a vector.
- int [vsum](#) (std::vector< int > const &data)
Sum the elements of a vector.
- std::vector< double > [operator*](#) (std::vector< double > const &data, double constant)
Multiplication of a vector with a scalar.
- std::vector< double > [operator/](#) (std::vector< double > const &data, double constant)
Division of a vector with a scalar.
- std::vector< double > [operator*](#) (std::vector< double > const &data1, std::vector< double > const &data2)
Elementwise multiplication of two vectors.
- std::vector< double > [operator+](#) (std::vector< double > const &data1, std::vector< double > const &data2)
Elementwise addition of two vectors.
- std::vector< double > [operator-](#) (std::vector< double > const &data1, std::vector< double > const &data2)
Elementwise subtraction of two vectors.
- std::vector< int > [operator-](#) (std::vector< int > const &data1, std::vector< int > const &data2)
Elementwise subtraction of two vectors.
- void [vadd](#) (std::vector< double > &data1, std::vector< double > const &data2)
Elementwise addition of two vectors.
- void [vadd](#) (std::vector< int > &data1, std::vector< int > const &data2)
Elementwise addition of two vectors.
- void [vsub](#) (std::vector< double > &data1, std::vector< double > const &data2)
Elementwise subtraction of two vectors.

- void **vsub** (std::vector< int > &data1, std::vector< int > const &data2)
Elementwise subtraction of two vectors.
- void **vsquare** (std::vector< double > &data1)
Elementwise square of a vector.
- void **vnormalize** (std::vector< double > &data1, std::vector< double > const &data2)
Normalize a vector to the same norm as another vector. Vectors are in this function treated as histograms, and the 'norm' is to be understood as the area of the histogram, here the sum of the vector elements. This procedure will not work well for negative valued vectors.

4.10.1 Detailed Description

File containing implementations of the utility functions defined in [mathutils.h](#).

Definition in file [mathutils.cpp](#).

4.10.2 Function Documentation

4.10.2.1 double calculate_distance (const double point1[3], const double point2[3])

Function to calculate the distance between two points.

Parameters

<i>point1</i>	The first point.
<i>point2</i>	The second point.

Returns

The distance between the two points.

Definition at line 32 of file mathutils.cpp.

4.10.2.2 std::vector<double> operator* (std::vector< double > const & data, double constant)

Multiplication of a vector with a scalar.

Parameters

<i>data</i>	: The vector to multiply.
<i>constant</i>	: The scalar to multiply with.

Returns

: A vector where all values are scaled with the constant.

Definition at line 103 of file mathutils.cpp.

4.10.2.3 `std::vector<double> operator*(std::vector< double > const & data1, std::vector< double > const & data2)`

Elementwise multiplication of two vectors.

Parameters

<i>data1</i>	: The first vector.
<i>data2</i>	: The second vector.

Returns

: A vector where all values are the elementwise products of the input vectors.

Definition at line 135 of file mathutils.cpp.

4.10.2.4 `std::vector<double> operator+ (std::vector< double > const & data1, std::vector< double > const & data2)`

Elementwise addition of two vectors.

Parameters

<i>data1</i>	: The first vector.
<i>data2</i>	: The second vector.

Returns

: The elementwise sum vector.

Definition at line 150 of file mathutils.cpp.

4.10.2.5 `std::vector<double> operator- (std::vector< double > const & data1, std::vector< double > const & data2)`

Elementwise subtraction of two vectors.

Parameters

<i>data1</i>	: The first vector.
<i>data2</i>	: The second vector.

Returns

: The elementwise difference vector.

Definition at line 173 of file mathutils.cpp.

4.10.2.6 `std::vector<int> operator- (std::vector< int > const & data1, std::vector< int > const & data2)`

Elementwise subtraction of two vectors.

Parameters

<i>data1</i>	: The first vector.
<i>data2</i>	: The second vector.

Returns

: The elementwise difference vector.

Definition at line 197 of file mathutils.cpp.

4.10.2.7 `std::vector<double> operator/ (std::vector< double > const & data, double constant)`

Division of a vector with a scalar.

Parameters

<i>data</i>	: The vector to divide.
<i>constant</i>	: The scalar to divide by.

Returns

: A vector where all values are scaled with inverse of the constant.

Definition at line 125 of file mathutils.cpp.

4.10.2.8 `Matrix setup_index_matrix (const int ntypes)`

Generate the index matrix that maps partials to array indexed.

Parameters

<i>ntypes</i>	: The number of atom types.
---------------	-----------------------------

Returns

The index matrix.

Definition at line 40 of file mathutils.cpp.

4.10.2.9 void vadd (std::vector< double > & *data1*, std::vector< double > const & *data2*)

Elementwise addition of two vectors.

Parameters

<i>data1</i>	: The vector to wich the other vector is added.
<i>data2</i>	: The vector to add.

Definition at line 221 of file mathutils.cpp.

4.10.2.10 void vadd (std::vector< int > & *data1*, std::vector< int > const & *data2*)

Elementwise addition of two vectors.

Parameters

<i>data1</i>	: The vector to wich the other vector is added.
<i>data2</i>	: The vector to add.

Definition at line 238 of file mathutils.cpp.

4.10.2.11 void vnormalize (std::vector< double > & *data1*, std::vector< double > const & *data2*)

Normalize a vector to the same norm as another vector. Vectors are in this function treated as histograms, and the 'norm' is to be understood as the area of the histogram, here the sum of the vector elements. This procedure will not work well for negative valued vectors.

Parameters

<i>data1</i>	: The vector to normalize.
<i>data2</i>	: The vector that provides the norm to use.

Definition at line 308 of file mathutils.cpp.

4.10.2.12 void vsquare (std::vector< double > & *data1*)

Elementwise square of a vector.

Parameters

<i>data</i>	: The vector to square.
-------------	-------------------------

Definition at line 292 of file mathutils.cpp.

4.10.2.13 void vsub (std::vector< double > & data1, std::vector< double > const & data2)

Elementwise subtraction of two vectors.

Parameters

<i>data1</i>	: The vector from wich the other vector is subtracted.
<i>data2</i>	: The vector to subtract.

Definition at line 256 of file mathutils.cpp.

4.10.2.14 void vsub (std::vector< int > & data1, std::vector< int > const & data2)

Elementwise subtraction of two vectors.

Parameters

<i>data1</i>	: The vector from wich the other vector is subtracted.
<i>data2</i>	: The vector to subtract.

Definition at line 274 of file mathutils.cpp.

4.10.2.15 double vsum (std::vector< double > const & data)

Sum the elements of a vector.

Parameters

<i>data</i>	: The vector to sum.
-------------	----------------------

Returns

: The sum of the vector elements.

Definition at line 63 of file mathutils.cpp.

4.10.2.16 int vsum (std::vector< int > const & data)

Sum the elements of a vector.

Parameters

<i>data</i>	: The vector to sum.
-------------	----------------------

Returns

: The sum of the vector elements.

Definition at line 88 of file mathutils.cpp.

4.11 mathutils.h File Reference

File containing definitions of utility functions and constants for mathematical operation.

```
#include <vector>
```

Functions

- double [fastfloor](#) (const double f)
Fast floor implementation as suggested by shark (the mac profiling program)
- double [calculate_distance](#) (const double point1[3], const double point2[3])
Function to calculate the distance between two points.
- [Matrix setup_index_matrix](#) (const int ntypes)
Generate the index matrix that maps partials to array indiceds.
- double [vsum](#) (std::vector< double > const &data)
Sum the elements of a vector.
- int [vsum](#) (std::vector< int > const &data)
Sum the elements of a vector.
- std::vector< double > [operator*](#) (std::vector< double > const &data, double constant)
Multiplication of a vector with a scalar.
- std::vector< double > [operator/](#) (std::vector< double > const &data, double constant)
Division of a vector with a scalar.
- std::vector< double > [operator*](#) (std::vector< double > const &data1, std::vector< double > const &data2)
Elementwise multiplication of two vectors.
- std::vector< double > [operator+](#) (std::vector< double > const &data1, std::vector< double > const &data2)
Elementwise addition of two vectors.
- std::vector< double > [operator-](#) (std::vector< double > const &data1, std::vector< double > const &data2)
Elementwise subtraction of two vectors.
- std::vector< int > [operator-](#) (std::vector< int > const &data1, std::vector< int > const &data2)
Elementwise subtraction of two vectors.

- void [vadd](#) (std::vector< double > &data1, std::vector< double > const &data2)
Elementwise addition of two vectors.
- void [vadd](#) (std::vector< int > &data1, std::vector< int > const &data2)
Elementwise addition of two vectors.
- void [vsub](#) (std::vector< double > &data1, std::vector< double > const &data2)
Elementwise subtraction of two vectors.
- void [vsub](#) (std::vector< int > &data1, std::vector< int > const &data2)
Elementwise subtraction of two vectors.
- void [vsquare](#) (std::vector< double > &data1)
Elementwise square of a vector.
- void [vnormalize](#) (std::vector< double > &data1, std::vector< double > const &data2)
Normalize a vector to the same norm as another vector. Vectors are in this function treated as histograms, and the 'norm' is to be understood as the area of the histogram, here the sum of the vector elements. This procedure will not work well for negative valued vectors.

Variables

- const double [PI__](#) = 3.1415926535897932384626433832795028841971693993
Global definition of PI.

4.11.1 Detailed Description

File containing definitions of utility functions and constants for mathematical operation.

Definition in file [mathutils.h](#).

4.11.2 Function Documentation

4.11.2.1 double calculate_distance (const double *point1*[3], const double *point2*[3])

Function to calculate the distance between two points.

Parameters

<i>point1</i>	The first point.
<i>point2</i>	The second point.

Returns

The distance between the two points.

Definition at line 32 of file mathutils.cpp.

4.11.2.2 double fastfloor (const double *f*) [inline]

Fast floor implementation as suggested by shark (the mac profiling program)

Parameters

<i>f</i>	: The double to floor.
----------	------------------------

Returns

: The floor value, i.e. the integer part of the value.

Definition at line 35 of file mathutils.h.

4.11.2.3 std::vector<double> operator* (std::vector< double > const & *data1*, std::vector< double > const & *data2*)

Elementwise multiplication of two vectors.

Parameters

<i>data1</i>	: The first vector.
<i>data2</i>	: The second vector.

Returns

: A vector where all values are the elementwise products of the input vectors.

Definition at line 135 of file mathutils.cpp.

4.11.2.4 std::vector<double> operator* (std::vector< double > const & *data*, double *constant*)

Multiplication of a vector with a scalar.

Parameters

<i>data</i>	: The vector to multiply.
<i>constant</i>	: The scalar to multiply with.

Returns

: A vector where all values are scaled with the constant.

Definition at line 103 of file mathutils.cpp.

4.11.2.5 `std::vector<double> operator+ (std::vector< double > const & data1, std::vector< double > const & data2)`

Elementwise addition of two vectors.

Parameters

<i>data1</i>	: The first vector.
<i>data2</i>	: The second vector.

Returns

: The elementwise sum vector.

Definition at line 150 of file mathutils.cpp.

4.11.2.6 `std::vector<double> operator- (std::vector< double > const & data1, std::vector< double > const & data2)`

Elementwise subtraction of two vectors.

Parameters

<i>data1</i>	: The first vector.
<i>data2</i>	: The second vector.

Returns

: The elementwise difference vector.

Definition at line 173 of file mathutils.cpp.

4.11.2.7 `std::vector<int> operator- (std::vector< int > const & data1, std::vector< int > const & data2)`

Elementwise subtraction of two vectors.

Parameters

<i>data1</i>	: The first vector.
<i>data2</i>	: The second vector.

Returns

: The elementwise difference vector.

Definition at line 197 of file mathutils.cpp.

4.11.2.8 `std::vector<double> operator/ (std::vector< double > const & data, double constant)`

Division of a vector with a scalar.

Parameters

<i>data</i>	: The vector to divide.
<i>constant</i>	: The scalar to divide by.

Returns

: A vector where all values are scaled with inverse of the constant.

Definition at line 125 of file mathutils.cpp.

4.11.2.9 `Matrix setup_index_matrix (const int ntypes)`

Generate the index matrix that maps partials to array indices.

Parameters

<i>ntypes</i>	: The number of atom types.
---------------	-----------------------------

Returns

The index matrix.

Definition at line 40 of file mathutils.cpp.

4.11.2.10 `void vadd (std::vector< double > & data1, std::vector< double > const & data2)`

Elementwise addition of two vectors.

Parameters

<i>data1</i>	: The vector to which the other vector is added.
<i>data2</i>	: The vector to add.

Definition at line 221 of file mathutils.cpp.

4.11.2.11 `void vadd (std::vector< int > & data1, std::vector< int > const & data2)`

Elementwise addition of two vectors.

Parameters

<i>data1</i>	: The vector to which the other vector is added.
<i>data2</i>	: The vector to add.

Definition at line 238 of file mathutils.cpp.

4.11.2.12 `void vnormalize (std::vector< double > & data1, std::vector< double > const & data2)`

Normalize a vector to the same norm as another vector. Vectors are in this function treated as histograms, and the 'norm' is to be understood as the area of the histogram, here the sum of the vector elements. This procedure will not work well for negative valued vectors.

Parameters

<i>data1</i>	: The vector to normalize.
<i>data2</i>	: The vector that provides the norm to use.

Definition at line 308 of file mathutils.cpp.

4.11.2.13 `void vsquare (std::vector< double > & data1)`

Elementwise square of a vector.

Parameters

<i>data</i>	: The vector to square.
-------------	-------------------------

Definition at line 292 of file mathutils.cpp.

4.11.2.14 `void vsub (std::vector< int > & data1, std::vector< int > const & data2)`

Elementwise subtraction of two vectors.

Parameters

<i>data1</i>	: The vector from which the other vector is subtracted.
<i>data2</i>	: The vector to subtract.

Definition at line 274 of file mathutils.cpp.

4.11.2.15 `void vsub (std::vector< double > & data1, std::vector< double > const & data2)`

Elementwise subtraction of two vectors.

Parameters

<i>data1</i>	: The vector from which the other vector is subtracted.
<i>data2</i>	: The vector to subtract.

Definition at line 256 of file mathutils.cpp.

4.11.2.16 `double vsum (std::vector< double > const & data)`

Sum the elements of a vector.

Parameters

<code>data</code>	: The vector to sum.
-------------------	----------------------

Returns

: The sum of the vector elements.

Definition at line 63 of file mathutils.cpp.

4.11.2.17 `int vsum (std::vector< int > const & data)`

Sum the elements of a vector.

Parameters

<code>data</code>	: The vector to sum.
-------------------	----------------------

Returns

: The sum of the vector elements.

Definition at line 88 of file mathutils.cpp.

4.11.3 Variable Documentation

4.11.3.1 `const double PI_ = 3.1415926535897932384626433832795028841971693993`

Global definition of PI.

Definition at line 28 of file mathutils.h.

4.12 `matrix.cpp` File Reference

File for the [Matrix](#) class implementation.

```
#include "matrix.h"
```

4.12.1 Detailed Description

File for the [Matrix](#) class implementation.

Definition in file [matrix.cpp](#).

4.13 matrix.h File Reference

File for the [Matrix](#) class definition.

```
#include <vector>
```

Classes

- class [Matrix](#)
Class for representing a 2D double matrix.

4.13.1 Detailed Description

File for the [Matrix](#) class definition.

Definition in file [matrix.h](#).

4.14 meanscalardata.cpp File Reference

File for the [MeanScalarData](#) class definition.

```
#include <cstdio>
#include <iostream>
#include <fstream>
#include "ioutils.h"
#include "library.h"
#include "meanscalardata.h"
```

4.14.1 Detailed Description

File for the [MeanScalarData](#) class definition.

Definition in file [meanscalardata.cpp](#).

4.15 meanscalardata.h File Reference

File for the [MeanScalarData](#) class definition.

```
#include <string>
```

```
#include <vector>
#include "basiscontainer.h"
```

Classes

- class [MeanScalarData](#)
Class for representing a mean scalar data set.

4.15.1 Detailed Description

File for the [MeanScalarData](#) class definition.

Definition in file [meanscalardata.h](#).

4.16 mklib.cpp File Reference

File for the [Mklib](#) class implementation.

```
#include <fstream>
#include <iostream>
#include <vector>
#include <cstdio>
#include <cstdlib>
#include "mklib.h"
#include "ioutils.h"
#include "ioexception.h"
```

4.16.1 Detailed Description

File for the [Mklib](#) class implementation.

Definition in file [mklib.cpp](#).

4.17 mklib.h File Reference

File for the [Mklib](#) class definition.

```
#include <string>
#include <vector>
#include <iosfwd>
```



```
#include "library.h"
```

Classes

- class [Mklib](#)

Class for implementing the main functionality of the mklib program.

4.17.1 Detailed Description

File for the [Mklib](#) class definition.

Definition in file [mklib.h](#).

4.18 mklibmain.cpp File Reference

File for the implementation of a simple command line interface to the [Mklib](#) library compiler utility.

```
#include <string>
#include <stdio>
#include "mklib.h"
```

Functions

- void [print_help](#) ()
- void [print_version](#) ()
- int [main](#) (const int argc, const char *argv[])

4.18.1 Detailed Description

File for the implementation of a simple command line interface to the [Mklib](#) library compiler utility.

Definition in file [mklibmain.cpp](#).

4.18.2 Function Documentation

4.18.2.1 int main (const int *argc*, const char * *argv*[])

Definition at line 58 of file [mklibmain.cpp](#).

4.18.2.2 void print_help ()

Definition at line 31 of file mklibmain.cpp.

4.18.2.3 void print_version ()

Definition at line 50 of file mklibmain.cpp.

4.19 mlrmc.cpp File Reference

File for the [Mlrmc](#) class implementation.

```
#include <fstream>
#include <iostream>
#include <vector>
#include <cstdio>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include "ioutils.h"
#include "mlrnd.h"
#include "mlrmc.h"
```

4.19.1 Detailed Description

File for the [Mlrmc](#) class implementation.

Definition in file [mlrmc.cpp](#).

4.20 mlrmc.h File Reference

File for the [Mlrmc](#) class definition.

```
#include <vector>
#include <string>
#include <fstream>
#include "specswap.h"
#include "mlrnd.h"
#include <iosfwd>
```

Classes

- class [Mlrnc](#)

The central RMC driver class, handling input setup and program flow.

4.20.1 Detailed Description

File for the [Mlrnc](#) class definition.

Definition in file [mlrnc.h](#).

4.21 mlrnd.cpp File Reference

File for the [Mlrnd](#) random number generator class implementation.

```
#include <cstdlib>
#include <cstdio>
#include "mlrnd.h"
#include "randf.h"
```

4.21.1 Detailed Description

File for the [Mlrnd](#) random number generator class implementation.

Definition in file [mlrnd.cpp](#).

4.22 mlrnd.h File Reference

File for the [Mlrnd](#) random number generator class definition.

Classes

- class [Mlrnd](#)

Class for representing the random number generator.

4.22.1 Detailed Description

File for the [Mlrnd](#) random number generator class definition.

Definition in file [mlrnd.h](#).

4.23 pcfdata.cpp File Reference

File for the [PCFData](#) class implementation.

```
#include <fstream>
#include <cmath>
#include "ioutils.h"
#include "mathutils.h"
#include "library.h"
#include "pcfdata.h"
```

4.23.1 Detailed Description

File for the [PCFData](#) class implementation.

Definition in file [pcfdata.cpp](#).

4.24 pcfdata.h File Reference

File for the [PCFData](#) class definition.

```
#include <vector>
#include <string>
#include "basiscontainer.h"
```

Classes

- class [PCFData](#)
Class representing a PCF data set.

4.24.1 Detailed Description

File for the [PCFData](#) class definition.

Definition in file [pcfdata.h](#).

4.25 randf.f File Reference

Functions

- subroutine [RND1](#) (S, ANS)

4.25.1 Function Documentation

4.25.1.1 subroutine RND1 (INTEGER S, DOUBLE PRECISION ANS)

Definition at line 8 of file randf.f.

4.26 randf.h File Reference

File for a C declaration of the FORTRAN rnd1 routine.

Functions

- void [rnd1_](#) (int &s, double &ans)

Declaration of the fortran function.

4.26.1 Detailed Description

File for a C declaration of the FORTRAN rnd1 routine.

Definition in file [randf.h](#).

4.26.2 Function Documentation

4.26.2.1 void rnd1_ (int & s, double & ans)

Declaration of the fortran function.

4.27 sampleset.cpp File Reference

File for the [Sampleset](#) class implementation.

```
#include <algorithm>
#include <fstream>
#include <cstdlib>
#include "sampleset.h"
#include "mathutils.h"
#include "ioutils.h"
```

4.27.1 Detailed Description

File for the [Sampleset](#) class implementation.

Definition in file [sampleset.cpp](#).

4.28 sampleset.h File Reference

File for the [Sampleset](#) class definition.

```
#include <vector>
#include <string>
```

Classes

- class [Sampleset](#)

Class representing the [Sampleset](#) in the SpecSwap simulation, for book keeping indices and performing swap moves.

4.28.1 Detailed Description

File for the [Sampleset](#) class definition.

Definition in file [sampleset.h](#).

4.29 scalardistributiondata.cpp File Reference

File for the [ScalarDistributionData](#) class implementation.

```
#include <fstream>
#include <cstdlib>
#include "library.h"
#include "ioutils.h"
#include "mathutils.h"
#include "scalardistributiondata.h"
```

4.29.1 Detailed Description

File for the [ScalarDistributionData](#) class implementation.

Definition in file [scalardistributiondata.cpp](#).

4.30 `scaldistributiondata.h` File Reference

File for the [ScalarDistributionData](#) class definition.

```
#include <string>
#include <vector>
#include "basiscontainer.h"
```

Classes

- class [ScalarDistributionData](#)
Class for representing a scalar distribution data set.

4.30.1 Detailed Description

File for the [ScalarDistributionData](#) class definition.

Definition in file [scaldistributiondata.h](#).

4.31 `specswap.cpp` File Reference

File for the [Specswap](#) class implementation.

```
#include <cstdlib>
#include <cmath>
#include <fstream>
#include <stdexcept>
#include "specswap.h"
#include "mathutils.h"
#include "ioutils.h"
```

4.31.1 Detailed Description

File for the [Specswap](#) class implementation.

Definition in file [specswap.cpp](#).

4.32 `specswap.h` File Reference

File for the [Specswap](#) class definition.

```
#include <vector>
#include <string>
#include "library.h"
#include "sampleset.h"
#include "mlrnd.h"
#include "pcfdata.h"
#include "curvedata.h"
#include "meanscalardata.h"
#include "valuescalardata.h"
#include "scalardistributiondata.h"
#include "basiscontainer.h"
#include <iosfwd>
```

Classes

- class [Specswap](#)

The central [Specswap](#) workhorse object performing moves and controlling communication with attached data sets.

4.32.1 Detailed Description

File for the [Specswap](#) class definition.

Definition in file [specswap.h](#).

4.33 specswapmain.cpp File Reference

File for the SpecSwap program's main routine.

```
#include <stdio.h>
#include "mlrmc.h"
#include "ioutils.h"
#include "ioexception.h"
```

Functions

- int [main](#) (int argc, char *argv[])

The main routine of the SpecSwap-RMC program.

4.33.1 Detailed Description

File for the SpecSwap program's main routine.

Definition in file [specswapmain.cpp](#).

4.33.2 Function Documentation

4.33.2.1 `int main (int argc, char * argv[])`

The main routine of the SpecSwap-RMC program.

Definition at line 29 of file [specswapmain.cpp](#).

4.34 valuescalardata.cpp File Reference

File for the [ValueScalarData](#) class implementation.

```
#include <cstdio>
#include <iostream>
#include <fstream>
#include "library.h"
#include "ioutils.h"
#include "valuescalardata.h"
```

4.34.1 Detailed Description

File for the [ValueScalarData](#) class implementation.

Definition in file [valuescalardata.cpp](#).

4.35 valuescalardata.h File Reference

File for the [ValueScalarData](#) class definition.

```
#include <string>
#include <vector>
#include "basiscontainer.h"
```

Classes

- class [ValueScalarData](#)

Class representing a value scalar data set.

4.35.1 Detailed Description

File for the [ValueScalarData](#) class definition.

Definition in file [valuescalardata.h](#).

Index

- ~IOException
 - IOException, [17](#)
- accept
 - CurveData, [10](#)
 - MeanScalarData, [44](#)
 - Mrmc, [54](#)
 - PCFData, [66](#)
 - ScalarDistributionData, [77](#)
 - Specswap, [87](#)
 - ValueScalarData, [99](#)
- accepted_
 - Mrmc, [57](#)
- accepted_recent_dump_
 - Mrmc, [57](#)
- accepted_recent_print_
 - Mrmc, [57](#)
- accepted_recent_probe_
 - Mrmc, [58](#)
- add_base
 - Library, [24](#)
- add_convolute
 - Library, [24](#)
- add_curve
 - Library, [25](#)
 - Specswap, [87](#)
- add_ending
 - Library, [25](#)
- add_format
 - Library, [25](#)
- add_geometry
 - Library, [25](#)
- add_index
 - Sampleset, [73](#)
- add_pcf
 - Specswap, [88](#)
- add_scalar_distribution
 - Specswap, [88](#)
- add_scalar_mean
 - Specswap, [88](#)
- add_scalar_name
 - Library, [25](#)
- add_scalar_value
 - Specswap, [89](#)
- add_scalars
 - Library, [26](#)
- add_scale
 - Library, [26](#)
- analyse_chunk
 - Specswap, [89](#)
- analysis_chunk_size_
 - Mrmc, [58](#)
- analysis_chunks_
 - Mrmc, [58](#)
- ANALYSIS_section_
 - Mrmc, [58](#)
- area_renorm_
 - CurveData, [13](#)
- atoms_per_type_
 - Library, [34](#)
- atomtypes_
 - Library, [34](#)
- attempted_
 - Mrmc, [58](#)
- attempted_recent_print_
 - Mrmc, [58](#)
- base_
 - Library, [34](#)
- BasisContainer, [5](#)
 - compareIndex, [6](#)
 - compareWeight, [6](#)
 - index, [6](#)
 - name, [6](#)
 - weight, [6](#)
- basiscontainer.h, [103](#)
- calc_scalar_pos
 - Specswap, [89](#)
- calculate_bin
 - PCFData, [66](#)
- calculate_chi2

- CurveData, 10
- MeanScalarData, 44
- PCFData, 66
- ScalarDistributionData, 77
- ValueScalarData, 99
- calculate_distance
 - mathutils.cpp, 123
 - mathutils.h, 129
- calculate_internals
 - Library, 26
- calculate_partial_histogram
 - PCFData, 67
- calculate_partials
 - Library, 26
- central_mol_
 - Library, 34
- central_molecule_
 - Library, 34
- check_eof
 - ioutils.cpp, 107
 - ioutils.h, 115
- check_keyword
 - Mklib, 48
- check_path
 - ioutils.cpp, 107
 - ioutils.h, 115
- check_positive_integer
 - ioutils.cpp, 108
 - ioutils.h, 115
- check_setup
 - Library, 26
- check_sigma
 - ioutils.cpp, 108
 - ioutils.h, 116
- chi2_
 - CurveData, 13
 - MeanScalarData, 46
 - Mrmc, 58
 - PCFData, 69
 - ScalarDistributionData, 80
 - Specswap, 94
 - ValueScalarData, 101
- chi2_new_
 - CurveData, 13
 - MeanScalarData, 46
 - Mrmc, 58
 - PCFData, 69
 - ScalarDistributionData, 80
 - Specswap, 94
 - ValueScalarData, 101
- chunk_analysis
 - Specswap, 89
- collect_weights
 - Specswap, 90
- columns_
 - Matrix, 41
- compareIndex
 - BasisContainer, 6
- compareWeight
 - BasisContainer, 6
- compile_library
 - Mklib, 49
- complete_setup
 - Library, 27
- Convolute, 7
 - no1, 7
 - no2, 7
 - no3, 7
 - no4, 7
- curve_
 - CurveData, 13
- curve_data_
 - Specswap, 94
- curve_name_
 - CurveData, 13
- curve_new_
 - CurveData, 13
- curve_reference_
 - CurveData, 14
- CurveData, 8
 - accept, 10
 - area_renorm_, 13
 - calculate_chi2, 10
 - chi2_, 13
 - chi2_new_, 13
 - curve_, 13
 - curve_name_, 13
 - curve_new_, 13
 - curve_reference_, 14
 - CurveData, 10
 - get_chi2, 11
 - get_chi2_new, 11
 - init, 11
 - notify, 11
 - nsample_, 14
 - one_over_sigma2_, 14
 - pos_, 14
 - print, 12
 - print_to_file, 12
 - read_reference, 12

- scale_, 14
- sigma_, 14
- Test_CurveData, 13
- weighted_analysis, 12
- curvedata.cpp, 103
- curvedata.h, 104
- data_
 - Matrix, 41
- debug_
 - Mklib, 50
- delta_chi2_
 - MIrmc, 59
- dimension_
 - Library, 34
- distribution_
 - ScalarDistributionData, 80
- distribution_new_
 - ScalarDistributionData, 80
- dr_
 - PCFData, 69
- dump_counter_
 - Specswap, 94
- dump_interval_
 - MIrmc, 59
- eatline
 - ioutils.cpp, 108
 - ioutils.h, 116
- empty_file_error
 - ioutils.cpp, 108
 - ioutils.h, 116
- endings_
 - Library, 34
- eof
 - ioutils.cpp, 108
 - ioutils.h, 116
- error
 - ioutils.cpp, 109
 - ioutils.h, 117
- error_exit
 - ioutils.cpp, 109
 - ioutils.h, 117
- factor_
 - ScalarDistributionData, 80
- fastfloor
 - mathutils.h, 130
- first_print
 - MIrmc, 54
- fit_interval_
 - PCFData, 69
- Format, 15
 - start, 15
 - step, 15
 - stop, 15
- from_basis_
 - Specswap, 94
- from_binary
 - Library, 27
- from_sample_
 - Specswap, 94
- FUNCTION
 - ioutils.h, 115
- geo_
 - Library, 34
- get_at
 - Library, 27
- get_basis_name
 - Library, 27
- get_bin
 - ScalarDistributionData, 78
- get_central
 - Library, 28
- get_chi2
 - CurveData, 11
 - MeanScalarData, 44
 - PCFData, 67
 - ScalarDistributionData, 78
 - Specswap, 90
 - ValueScalarData, 99
- get_chi2_new
 - CurveData, 11
 - MeanScalarData, 44
 - MIrmc, 54
 - PCFData, 67
 - ScalarDistributionData, 78
 - Specswap, 90
 - ValueScalarData, 100
- get_convolute
 - Library, 28
- get_ending
 - Library, 28
- get_format
 - Library, 28
- get_geo
 - Library, 29
- get_indices
 - Sampleset, 73

- get_internal
 - Library, [29](#)
- get_lib_version
 - Mklib, [49](#)
- get_name
 - Library, [29](#)
- get_nbase
 - Library, [30](#)
- get_ncurves
 - Library, [30](#)
- get_nscalars
 - Library, [30](#)
- get_partial
 - Library, [30](#)
- get_r
 - PCFData, [67](#)
- get_scalar_at
 - Library, [31](#)
- get_scalar_name
 - Library, [31](#)
- get_scale
 - Library, [31](#)
- highest_
 - ScalarDistributionData, [80](#)
- histogram_
 - PCFData, [70](#)
- histogram_new_
 - PCFData, [70](#)
- index
 - BasisContainer, [6](#)
- index_at
 - Sampleset, [73](#)
- index_matrix_
 - Library, [35](#)
- indices_
 - Sampleset, [74](#)
- info
 - IOException, [17](#)
- init
 - CurveData, [11](#)
 - MeanScalarData, [44](#)
 - PCFData, [68](#)
 - ScalarDistributionData, [78](#)
 - ValueScalarData, [100](#)
- init_chi2
 - Mrmc, [54](#)
- internals_
 - Library, [35](#)
- interval_
 - ValueScalarData, [101](#)
- IOException, [15](#)
 - ~IOException, [17](#)
 - info, [17](#)
 - IOException, [17](#)
 - location, [17](#)
 - location_, [18](#)
 - message, [17](#)
 - message_, [18](#)
 - print, [17](#)
 - what, [17](#)
- ioexception.cpp, [104](#)
- ioexception.h, [104](#)
- ioutils.cpp, [105](#)
 - check_eof, [107](#)
 - check_path, [107](#)
 - check_positive_integer, [108](#)
 - check_sigma, [108](#)
 - eatline, [108](#)
 - empty_file_error, [108](#)
 - eof, [108](#)
 - error, [109](#)
 - error_exit, [109](#)
 - missing_keyword_error, [109](#)
 - newline_indent, [109](#)
 - open_file_error, [110](#)
 - print_startup, [110](#)
 - print_success, [110](#)
 - read_keyword_error, [110](#)
 - same_keyword_error, [111](#)
 - same_section_error, [111](#)
 - start_timer, [111](#)
 - time_0__, [112](#)
 - timer, [111](#)
 - timestamp, [111](#)
 - to_string, [111](#)
 - unknown_keyword_error, [112](#)
 - unknown_section_error, [112](#)
- ioutils.h, [112](#)
 - check_eof, [115](#)
 - check_path, [115](#)
 - check_positive_integer, [115](#)
 - check_sigma, [116](#)
 - eatline, [116](#)
 - empty_file_error, [116](#)
 - eof, [116](#)
 - error, [117](#)
 - error_exit, [117](#)
 - FUNCTION, [115](#)

- LOCATION, 115
- missing_keyword_error, 117
- newline_indent, 117
- open_file_error, 118
- print_startup, 118
- print_success, 118
- read_keyword_error, 118
- same_keyword_error, 118
- same_section_error, 119
- start_timer, 119
- timer, 119
- timestamp, 119
- to_string, 119
- unknown_keyword_error, 120
- unknown_section_error, 120
- is_added
 - Sampleset, 73
- last_print
 - MIrmc, 54
- Library, 18
 - add_base, 24
 - add_convolute, 24
 - add_curve, 25
 - add_ending, 25
 - add_format, 25
 - add_geometry, 25
 - add_scalar_name, 25
 - add_scalars, 26
 - add_scale, 26
 - atoms_per_type_, 34
 - atomtypes_, 34
 - base_, 34
 - calculate_internals, 26
 - calculate_partials, 26
 - central_mol_, 34
 - central_molecule_, 34
 - check_setup, 26
 - complete_setup, 27
 - dimension_, 34
 - endings_, 34
 - from_binary, 27
 - geo_, 34
 - get_at, 27
 - get_basis_name, 27
 - get_central, 28
 - get_convolute, 28
 - get_ending, 28
 - get_format, 28
 - get_geo, 29
 - get_internal, 29
 - get_name, 29
 - get_nbase, 30
 - get_ncurves, 30
 - get_nscalars, 30
 - get_partial, 30
 - get_scalar_at, 31
 - get_scalar_name, 31
 - get_scale, 31
 - index_matrix_, 35
 - internals_, 35
 - Library, 24
 - nadded_, 35
 - nadded_scale_, 35
 - name_, 35
 - names_, 35
 - natoms_, 35
 - nbase_, 35
 - ncurves_, 36
 - no1_, 36
 - no2_, 36
 - no3_, 36
 - no4_, 36
 - npairs_, 36
 - nscalars_, 36
 - ntypes_, 36
 - partials_, 37
 - read_scale, 31
 - scalar_names_, 37
 - scalars_, 37
 - scale_, 37
 - set_atoms_info, 32
 - set_atomtypes, 32
 - set_name, 32
 - set_nbase, 32
 - set_ncurves, 33
 - set_nscalars, 33
 - setup_done_, 37
 - start_, 37
 - step_, 37
 - stop_, 37
 - Test_Library, 33
 - to_binary, 33
 - use_atoms_info_, 38
 - version_, 38
- library.cpp, 120
- library.h, 121
- library_
 - Specswap, 94
- library_from_info_file

- Mklib, 49
- LOCATION
 - ioutils.h, 115
- location
 - IOException, 17
- location_
 - IOException, 18
- logfile_
 - Mrmc, 59
- lowest_
 - ScalarDistributionData, 80
- main
 - mklibmain.cpp, 137
 - specswapmain.cpp, 145
- mathutils.cpp, 121
 - calculate_distance, 123
 - operator*, 123, 124
 - operator+, 124
 - operator-, 124, 125
 - operator/, 125
 - setup_index_matrix, 125
 - vadd, 125, 126
 - vnormalize, 126
 - vsquare, 126
 - vsub, 126, 127
 - vsum, 127
- mathutils.h, 128
 - calculate_distance, 129
 - fastfloor, 130
 - operator*, 130
 - operator+, 131
 - operator-, 131
 - operator/, 131
 - PI_, 134
 - setup_index_matrix, 132
 - vadd, 132
 - vnormalize, 133
 - vsquare, 133
 - vsub, 133
 - vsum, 134
- Matrix, 38
 - columns_, 41
 - data_, 41
 - Matrix, 39
 - operator(), 40
 - resize, 40
 - rows_, 41
 - Test_Matrix, 40
- matrix.cpp, 134
- matrix.h, 135
- mean_scalar_data_
 - Specswap, 95
- MeanScalarData, 41
 - accept, 44
 - calculate_chi2, 44
 - chi2_, 46
 - chi2_new_, 46
 - get_chi2, 44
 - get_chi2_new, 44
 - init, 44
 - MeanScalarData, 43
 - name_, 46
 - notify, 45
 - nsample_, 46
 - one_over_sigma2_, 46
 - pos_, 46
 - print, 45
 - sigma_, 46
 - target_, 46
 - Test_MeanScalarData, 45
 - value_, 47
 - value_new_, 47
 - weighted_analysis, 45
- meanscalardata.cpp, 135
- meanscalardata.h, 135
- message
 - IOException, 17
- message_
 - IOException, 18
- missing_keyword_error
 - ioutils.cpp, 109
 - ioutils.h, 117
- Mklib, 47
 - check_keyword, 48
 - compile_library, 49
 - debug_, 50
 - get_lib_version, 49
 - library_from_info_file, 49
 - Mklib, 48
 - Test_Mklib, 49
 - verbos, 50
- mklib.cpp, 136
- mklib.h, 136
- mklibmain.cpp, 137
 - main, 137
 - print_help, 137
 - print_version, 138
- Mrmc, 50
 - accept, 54

- accepted_, 57
- accepted_recent_dump_, 57
- accepted_recent_print_, 57
- accepted_recent_probe_, 58
- analysis_chunk_size_, 58
- analysis_chunks_, 58
- ANALYSIS_section_, 58
- attempted_, 58
- attempted_recent_print_, 58
- chi2_, 58
- chi2_new_, 58
- delta_chi2_, 59
- dump_interval_, 59
- first_print, 54
- get_chi2_new, 54
- init_chi2, 54
- last_print, 54
- logfile_, 59
- Mrmc, 54
- montecarlo_test, 55
- move, 55
- moves_, 59
- notify, 55
- post_process, 55
- print, 55
- print_and_save, 55
- print_interval_, 59
- print_post_process_start, 55
- print_post_process_stop, 55
- probe_counter_, 59
- probe_interval_, 59
- rand_, 59
- read_ANALYSIS, 56
- read_CURVE, 56
- read_DISTRIBUTION, 56
- read_MEAN, 56
- read_PCF, 56
- read_RUN, 56
- read_SCALAR, 56
- read_section, 56
- read_VALUE, 57
- rmc_loop, 57
- run_rmc, 57
- RUN_section_, 60
- save, 57
- save_interval_, 60
- seed_, 60
- specswap_, 60
- title_, 60
- mlrmc.cpp, 138
- mlrmc.h, 138
- Mrnd, 60
 - Mrnd, 61
 - random, 61
 - seed_, 62
 - set_seed, 62
 - Test_Mrnd, 62
- mlrnd.cpp, 139
- mlrnd.h, 139
- montecarlo_test
 - Mrmc, 55
- move
 - Mrmc, 55
 - Specswap, 90
- moves_
 - Mrmc, 59
- nadded_
 - Library, 35
- nadded_scale_
 - Library, 35
- name
 - BasisContainer, 6
- name_
 - Library, 35
 - MeanScalarData, 46
 - ScalarDistributionData, 81
 - ValueScalarData, 101
- names_
 - Library, 35
- natoms_
 - Library, 35
- nbase_
 - Library, 35
- nbins_
 - ScalarDistributionData, 81
- ncurves_
 - Library, 36
 - Specswap, 95
- newline_indent
 - ioutils.cpp, 109
 - ioutils.h, 117
- no1
 - Convolute, 7
- no1_
 - Library, 36
- no2
 - Convolute, 7
- no2_
 - Library, 36

- no3
 - Convolute, [7](#)
- no3_
 - Library, [36](#)
- no4
 - Convolute, [7](#)
- no4_
 - Library, [36](#)
- notify
 - CurveData, [11](#)
 - MeanScalarData, [45](#)
 - Mrmc, [55](#)
 - PCFData, [68](#)
 - ScalarDistributionData, [79](#)
 - Specswap, [91](#)
 - ValueScalarData, [100](#)
- npairs_
 - Library, [36](#)
- nprobe_
 - Specswap, [95](#)
- nsample_
 - CurveData, [14](#)
 - MeanScalarData, [46](#)
 - PCFData, [70](#)
 - Sampleset, [74](#)
 - ScalarDistributionData, [81](#)
 - Specswap, [95](#)
 - ValueScalarData, [101](#)
- nscalars_
 - Library, [36](#)
- ntypes_
 - Library, [36](#)
- numberdensity_
 - PCFData, [70](#)
- one_over_binsize_
 - ScalarDistributionData, [81](#)
- one_over_dr_
 - PCFData, [70](#)
- one_over_sigma2_
 - CurveData, [14](#)
 - MeanScalarData, [46](#)
 - PCFData, [70](#)
 - ScalarDistributionData, [81](#)
 - ValueScalarData, [102](#)
- open_file_error
 - ioutils.cpp, [110](#)
 - ioutils.h, [118](#)
- operator*
 - mathutils.cpp, [123](#), [124](#)
 - mathutils.h, [130](#)
- operator()
 - Matrix, [40](#)
- operator+
 - mathutils.cpp, [124](#)
 - mathutils.h, [131](#)
- operator-
 - mathutils.cpp, [124](#), [125](#)
 - mathutils.h, [131](#)
- operator/
 - mathutils.cpp, [125](#)
 - mathutils.h, [131](#)
- partial_
 - PCFData, [70](#)
- partials_
 - Library, [37](#)
- pcf_data_
 - Specswap, [95](#)
- pcf_normalization_factor_
 - PCFData, [70](#)
- pcf_reference_
 - PCFData, [71](#)
- PCFData, [62](#)
 - accept, [66](#)
 - calculate_bin, [66](#)
 - calculate_chi2, [66](#)
 - calculate_partial_histogram, [67](#)
 - chi2_, [69](#)
 - chi2_new_, [69](#)
 - dr_, [69](#)
 - fit_interval_, [69](#)
 - get_chi2, [67](#)
 - get_chi2_new, [67](#)
 - get_r, [67](#)
 - histogram_, [70](#)
 - histogram_new_, [70](#)
 - init, [68](#)
 - notify, [68](#)
 - nsample_, [70](#)
 - numberdensity_, [70](#)
 - one_over_dr_, [70](#)
 - one_over_sigma2_, [70](#)
 - partial_, [70](#)
 - pcf_normalization_factor_, [70](#)
 - pcf_reference_, [71](#)
 - PCFData, [65](#)
 - print, [68](#)
 - print_to_file, [68](#)
 - read_reference, [68](#)

- rmax_, 71
- rmin_, 71
- sigma_, 71
- Test_PCFData, 69
- weighted_analysis, 69
- pcfdata.cpp, 140
- pcfdata.h, 140
- PI__
 - mathutils.h, 134
- pos_
 - CurveData, 14
 - MeanScalarData, 46
 - ScalarDistributionData, 81
 - ValueScalarData, 102
- post_process
 - Mrmc, 55
- prepare_for_analysis
 - Specswap, 91
- print
 - CurveData, 12
 - IOException, 17
 - MeanScalarData, 45
 - Mrmc, 55
 - PCFData, 68
 - ScalarDistributionData, 79
 - Specswap, 91
 - ValueScalarData, 100
- print_and_save
 - Mrmc, 55
- print_help
 - mklibmain.cpp, 137
- print_interval_
 - Mrmc, 59
- print_post_process_start
 - Mrmc, 55
- print_post_process_stop
 - Mrmc, 55
- print_start
 - Specswap, 91
- print_startup
 - ioutils.cpp, 110
 - ioutils.h, 118
- print_stop
 - Specswap, 91
- print_success
 - ioutils.cpp, 110
 - ioutils.h, 118
- print_to_file
 - CurveData, 12
 - PCFData, 68
 - Specswap, 91
- print_version
 - mklibmain.cpp, 138
- print_weights
 - Specswap, 92
- probe_counter_
 - Mrmc, 59
- probe_interval_
 - Mrmc, 59
- rand_
 - Mrmc, 59
 - Specswap, 95
- randf.f, 140
 - RND1, 141
- randf.h, 141
 - rnd1_, 141
- random
 - Mrnd, 61
 - Specswap, 92
- random_basis
 - Specswap, 92
- read_ANALYSIS
 - Mrmc, 56
- read_CURVE
 - Mrmc, 56
- read_DISTRIBUTION
 - Mrmc, 56
- read_keyword_error
 - ioutils.cpp, 110
 - ioutils.h, 118
- read_MEAN
 - Mrmc, 56
- read_PCF
 - Mrmc, 56
- read_reference
 - CurveData, 12
 - PCFData, 68
 - ScalarDistributionData, 79
- read_RUN
 - Mrmc, 56
- read_SCALAR
 - Mrmc, 56
- read_scale
 - Library, 31
- read_section
 - Mrmc, 56
- read_VALUE
 - Mrmc, 57
- resize

- Matrix, [40](#)
- restart_
 - Specswap, [95](#)
- restart_path_
 - Specswap, [95](#)
- rmax_
 - PCFData, [71](#)
- rmc_loop
 - Mrmc, [57](#)
- rmin_
 - PCFData, [71](#)
- RND1
 - randf.f, [141](#)
- rnd1_
 - randf.h, [141](#)
- rows_
 - Matrix, [41](#)
- run_rmc
 - Mrmc, [57](#)
- RUN_section_
 - Mrmc, [60](#)
- same_keyword_error
 - ioutils.cpp, [111](#)
 - ioutils.h, [118](#)
- same_section_error
 - ioutils.cpp, [111](#)
 - ioutils.h, [119](#)
- Sampleset, [71](#)
 - add_index, [73](#)
 - get_indices, [73](#)
 - index_at, [73](#)
 - indices_, [74](#)
 - is_added, [73](#)
 - nsample_, [74](#)
 - Sampleset, [72](#)
 - swap_into_slot, [73](#)
 - Test_Sampleset, [74](#)
- sampleset.cpp, [141](#)
- sampleset.h, [142](#)
- sampleset_
 - Specswap, [96](#)
- save
 - Mrmc, [57](#)
- save_interval_
 - Mrmc, [60](#)
- scalar_distribution_data_
 - Specswap, [96](#)
- scalar_names_
 - Library, [37](#)
- ScalarDistributionData, [74](#)
 - accept, [77](#)
 - calculate_chi2, [77](#)
 - chi2_, [80](#)
 - chi2_new_, [80](#)
 - distribution_, [80](#)
 - distribution_new_, [80](#)
 - factor_, [80](#)
 - get_bin, [78](#)
 - get_chi2, [78](#)
 - get_chi2_new, [78](#)
 - highest_, [80](#)
 - init, [78](#)
 - lowest_, [80](#)
 - name_, [81](#)
 - nbins_, [81](#)
 - notify, [79](#)
 - nsample_, [81](#)
 - one_over_binsize_, [81](#)
 - one_over_sigma2_, [81](#)
 - pos_, [81](#)
 - print, [79](#)
 - read_reference, [79](#)
 - ScalarDistributionData, [77](#)
 - scale_, [81](#)
 - sigma_, [81](#)
 - target_, [82](#)
 - Test_ScalarDistributionData, [80](#)
 - weighted_analysis, [79](#)
- scaldistributiondata.cpp, [142](#)
- scaldistributiondata.h, [143](#)
- scalars_
 - Library, [37](#)
- scale_
 - CurveData, [14](#)
 - Library, [37](#)
 - ScalarDistributionData, [81](#)
- seed_
 - Mrmc, [60](#)
 - Mrnd, [62](#)
- set_atoms_info
 - Library, [32](#)
- set_atomtypes
 - Library, [32](#)
- set_name
 - Library, [32](#)
- set_nbase
 - Library, [32](#)
- set_ncurves
 - Library, [33](#)

- set_nscalars
 - Library, 33
- set_seed
 - Mrnd, 62
- setup
 - Specswap, 92
- setup_chi2
 - Specswap, 92
- setup_done_
 - Library, 37
- setup_index_matrix
 - mathutils.cpp, 125
 - mathutils.h, 132
- setup_sampleset
 - Specswap, 93
- sigma_
 - CurveData, 14
 - MeanScalarData, 46
 - PCFData, 71
 - ScalarDistributionData, 81
 - ValueScalarData, 102
- slot_
 - Specswap, 96
- Specswap, 82
 - accept, 87
 - add_curve, 87
 - add_pcf, 88
 - add_scalar_distribution, 88
 - add_scalar_mean, 88
 - add_scalar_value, 89
 - analyse_chunk, 89
 - calc_scalar_pos, 89
 - chi2_, 94
 - chi2_new_, 94
 - chunk_analysis, 89
 - collect_weights, 90
 - curve_data_, 94
 - dump_counter_, 94
 - from_basis_, 94
 - from_sample_, 94
 - get_chi2, 90
 - get_chi2_new, 90
 - library_, 94
 - mean_scalar_data_, 95
 - move, 90
 - ncurves_, 95
 - notify, 91
 - nprobe_, 95
 - nsample_, 95
 - pcf_data_, 95
 - prepare_for_analysis, 91
 - print, 91
 - print_start, 91
 - print_stop, 91
 - print_to_file, 91
 - print_weights, 92
 - rand_, 95
 - random, 92
 - random_basis, 92
 - restart_, 95
 - restart_path_, 95
 - sampleset_, 96
 - scalar_distribution_data_, 96
 - setup, 92
 - setup_chi2, 92
 - setup_sampleset, 93
 - slot_, 96
 - Specswap, 86, 87
 - Test_Specswap, 94
 - value_scalar_data_, 96
 - weighted_analysis, 93
 - weights_, 96
 - weights_table_, 96
 - write_dump, 93
 - write_restart, 93
 - write_weights_and_names_list, 93
- specswap.cpp, 143
- specswap.h, 143
- specswap_
 - Mrmc, 60
- specswapmain.cpp, 144
 - main, 145
- start
 - Format, 15
- start_
 - Library, 37
- start_timer
 - ioutils.cpp, 111
 - ioutils.h, 119
- step
 - Format, 15
- step_
 - Library, 37
- stop
 - Format, 15
- stop_
 - Library, 37
- swap_into_slot
 - Sampleset, 73

- target_
 - MeanScalarData, 46
 - ScalarDistributionData, 82
 - ValueScalarData, 102
- Test_CurveData
 - CurveData, 13
- Test_Library
 - Library, 33
- Test_Matrix
 - Matrix, 40
- Test_MeanScalarData
 - MeanScalarData, 45
- Test_Mklib
 - Mklib, 49
- Test_Mlrnd
 - Mlrnd, 62
- Test_PCFData
 - PCFData, 69
- Test_Sampleset
 - Sampleset, 74
- Test_ScalarDistributionData
 - ScalarDistributionData, 80
- Test_Specswap
 - Specswap, 94
- Test_ValueScalarData
 - ValueScalarData, 101
- time_0__
 - ioutils.cpp, 112
- timer
 - ioutils.cpp, 111
 - ioutils.h, 119
- timestamp
 - ioutils.cpp, 111
 - ioutils.h, 119
- title_
 - Mlrnc, 60
- to_binary
 - Library, 33
- to_string
 - ioutils.cpp, 111
 - ioutils.h, 119
- unknown_keyword_error
 - ioutils.cpp, 112
 - ioutils.h, 120
- unknown_section_error
 - ioutils.cpp, 112
 - ioutils.h, 120
- use_atoms_info_
 - Library, 38
- vadd
 - mathutils.cpp, 125, 126
 - mathutils.h, 132
- value_
 - MeanScalarData, 47
 - ValueScalarData, 102
- value_new_
 - MeanScalarData, 47
 - ValueScalarData, 102
- value_scalar_data_
 - Specswap, 96
- ValueScalarData, 97
 - accept, 99
 - calculate_chi2, 99
 - chi2_, 101
 - chi2_new_, 101
 - get_chi2, 99
 - get_chi2_new, 100
 - init, 100
 - interval_, 101
 - name_, 101
 - notify, 100
 - nsample_, 101
 - one_over_sigma2_, 102
 - pos_, 102
 - print, 100
 - sigma_, 102
 - target_, 102
 - Test_ValueScalarData, 101
 - value_, 102
 - value_new_, 102
 - ValueScalarData, 99
 - weighted_analysis, 100
- valuescalardata.cpp, 145
- valuescalardata.h, 145
- verbos
 - Mklib, 50
- version_
 - Library, 38
- vnormalize
 - mathutils.cpp, 126
 - mathutils.h, 133
- vsquare
 - mathutils.cpp, 126
 - mathutils.h, 133
- vsub
 - mathutils.cpp, 126, 127
 - mathutils.h, 133
- vsum
 - mathutils.cpp, 127

- mathutils.h, [134](#)
- weight
 - BasisContainer, [6](#)
- weighted_analysis
 - CurveData, [12](#)
 - MeanScalarData, [45](#)
 - PCFData, [69](#)
 - ScalarDistributionData, [79](#)
 - Specswap, [93](#)
 - ValueScalarData, [100](#)
- weights_
 - Specswap, [96](#)
- weights_table_
 - Specswap, [96](#)
- what
 - IOException, [17](#)
- write_dump
 - Specswap, [93](#)
- write_restart
 - Specswap, [93](#)
- write_weights_and_names_list
 - Specswap, [93](#)