

Metaboflux

2.0

Generated by Doxygen 1.7.4

Wed Apr 27 2011 16:40:50



# Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Data Structure Documentation</b>	<b>5</b>
3.1	Equations Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Field Documentation . . . . .	6
3.1.2.1	info . . . . .	6
3.1.2.2	type . . . . .	6
3.2	Equdata Union Reference . . . . .	6
3.2.1	Detailed Description . . . . .	6
3.2.2	Field Documentation . . . . .	6
3.2.2.1	data . . . . .	6
3.2.2.2	var . . . . .	7
3.3	Especies Struct Reference . . . . .	7
3.3.1	Detailed Description . . . . .	8
3.3.2	Field Documentation . . . . .	8
3.3.2.1	id . . . . .	8
3.3.2.2	quantite . . . . .	8
3.3.2.3	system . . . . .	8
3.4	ListArguments Struct Reference . . . . .	8
3.4.1	Detailed Description . . . . .	9
3.4.2	Field Documentation . . . . .	9

3.4.2.1	activity . . . . .	9
3.4.2.2	debug . . . . .	9
3.4.2.3	files_path . . . . .	9
3.4.2.4	group . . . . .	9
3.4.2.5	port . . . . .	9
3.5	ListParameters Struct Reference . . . . .	10
3.5.1	Detailed Description . . . . .	11
3.5.2	Field Documentation . . . . .	11
3.5.2.1	banned . . . . .	11
3.5.2.2	conf . . . . .	11
3.5.2.3	equation . . . . .	11
3.5.2.4	interest_parameters . . . . .	11
3.5.2.5	model . . . . .	11
3.5.2.6	nb_banned . . . . .	12
3.5.2.7	nb_couples . . . . .	12
3.5.2.8	nb_equations . . . . .	12
3.5.2.9	nb_parameters . . . . .	12
3.5.2.10	nb_reaction . . . . .	12
3.5.2.11	nb_triesMod . . . . .	13
3.5.2.12	nb_triesSa . . . . .	13
3.5.2.13	noeud . . . . .	13
3.5.2.14	parameters . . . . .	13
3.6	option Struct Reference . . . . .	13
3.6.1	Detailed Description . . . . .	13
3.7	Reaction Struct Reference . . . . .	14
3.7.1	Detailed Description . . . . .	14
3.7.2	Field Documentation . . . . .	14
3.7.2.1	link . . . . .	14
3.7.2.2	suivant . . . . .	14
3.8	Score Struct Reference . . . . .	15
3.8.1	Detailed Description . . . . .	15
3.8.2	Field Documentation . . . . .	15
3.8.2.1	name . . . . .	15
3.8.2.2	nb_reaction . . . . .	15

3.8.2.3	<code>nb_species</code>	16
3.8.2.4	<code>quantite</code>	16
3.8.2.5	<code>reaction</code>	16
3.8.2.6	<code>species</code>	16
3.8.2.7	<code>species_amount</code>	16
3.8.2.8	<code>species_weight</code>	17
3.8.2.9	<code>taille</code>	17
3.8.2.10	<code>tailleReactions</code>	17
3.8.2.11	<code>tailleSpecies</code>	17
3.9	<code>SimParameters</code> Struct Reference	17
3.9.1	Detailed Description	18
3.9.2	Field Documentation	18
3.9.2.1	<code>debugFile</code>	18
3.9.2.2	<code>out</code>	19
3.9.2.3	<code>pile</code>	19
3.9.2.4	<code>r</code>	19
3.9.2.5	<code>y</code>	19
3.10	<code>TestReaction</code> Struct Reference	19
3.10.1	Detailed Description	20
3.10.2	Field Documentation	20
3.10.2.1	<code>minStepTab</code>	20
3.10.2.2	<code>tabReactions</code>	20
3.11	<code>xmlConfig_t</code> Struct Reference	20
3.11.1	Detailed Description	21
3.11.2	Field Documentation	21
3.11.2.1	<code>ctxt</code>	21
3.11.2.2	<code>doc</code>	21
3.11.2.3	<code>fichier</code>	21
3.11.2.4	<code>racine</code>	21
<b>4</b>	<b>File Documentation</b>	<b>23</b>
4.1	<code>metaboflux/include/data_parameters.h</code> File Reference	23
4.1.1	Detailed Description	24
4.1.2	Function Documentation	25

4.1.2.1	Data_allocEquations . . . . .	25
4.1.2.2	Data_allocParameters . . . . .	26
4.1.2.3	Data_allocScore . . . . .	26
4.1.2.4	Data_allocSimParameters . . . . .	27
4.1.2.5	Data_copieTab . . . . .	28
4.1.2.6	Data_desallocation . . . . .	28
4.1.2.7	Data_desallocSim . . . . .	29
4.1.2.8	Data_equationsAlloc . . . . .	29
4.1.2.9	Data_equationsInit . . . . .	30
4.1.2.10	Data_initSBML . . . . .	31
4.1.2.11	Data_initXML . . . . .	32
4.1.2.12	Data_parameters . . . . .	32
4.1.2.13	Data_rngAlloc . . . . .	33
4.1.2.14	Data_scoreAlloc . . . . .	34
4.1.2.15	Data_scoreFree . . . . .	34
4.1.2.16	Data_scoreInit . . . . .	35
4.1.2.17	Data_simParameters . . . . .	35
4.1.2.18	Data_updateTab . . . . .	36
4.2	metaboflux/include/equations.h File Reference . . . . .	37
4.2.1	Detailed Description . . . . .	39
4.2.2	Enumeration Type Documentation . . . . .	39
4.2.2.1	Eqtype . . . . .	39
4.2.3	Function Documentation . . . . .	40
4.2.3.1	add . . . . .	40
4.2.3.2	divide . . . . .	40
4.2.3.3	equality . . . . .	40
4.2.3.4	Equations_addOp . . . . .	41
4.2.3.5	Equations_alloc . . . . .	42
4.2.3.6	Equations_calcul . . . . .	42
4.2.3.7	Equations_defiler . . . . .	44
4.2.3.8	Equations_defilerSuiv . . . . .	44
4.2.3.9	Equations_define . . . . .	44
4.2.3.10	Equations_depiler . . . . .	46
4.2.3.11	Equations_empiler . . . . .	46

4.2.3.12	Equations_emptyOp . . . . .	46
4.2.3.13	Equations_finalQuantite . . . . .	47
4.2.3.14	Equations_findSpecies . . . . .	48
4.2.3.15	Equations_pileFormation . . . . .	48
4.2.3.16	Equations_print . . . . .	51
4.2.3.17	Equations_priorite . . . . .	51
4.2.3.18	Equations_resultat . . . . .	51
4.2.3.19	Equations_sommet . . . . .	52
4.2.3.20	Equations_vide . . . . .	53
4.2.3.21	inf . . . . .	53
4.2.3.22	inf_equal . . . . .	53
4.2.3.23	multiply . . . . .	54
4.2.3.24	subtract . . . . .	54
4.2.3.25	sup . . . . .	54
4.2.3.26	sup_equal . . . . .	55
4.3	metaboflux/include/especies.h File Reference . . . . .	55
4.3.1	Detailed Description . . . . .	56
4.3.2	Function Documentation . . . . .	57
4.3.2.1	Especies_alloc . . . . .	57
4.3.2.2	Especies_allocReactions . . . . .	57
4.3.2.3	Especies_find . . . . .	58
4.3.2.4	Especies_free . . . . .	58
4.3.2.5	Especies_freeReactions . . . . .	59
4.3.2.6	Especies_getNbReactions . . . . .	59
4.3.2.7	Especies_getQuantite . . . . .	60
4.3.2.8	Especies_print . . . . .	60
4.3.2.9	Especies_print_2 . . . . .	60
4.3.2.10	Especies_save . . . . .	61
4.3.2.11	Especies_scoreSpecies . . . . .	61
4.3.2.12	Especies_setQuantite . . . . .	62
4.4	metaboflux/include/gsl_min.h File Reference . . . . .	62
4.4.1	Detailed Description . . . . .	63
4.4.2	Function Documentation . . . . .	63
4.4.2.1	Min_compute_minimization . . . . .	63

---

4.4.2.2	Min_copieTab2 . . . . .	65
4.4.2.3	Min_copieTab3 . . . . .	65
4.4.2.4	Min_getTampon . . . . .	65
4.4.2.5	Min_my_f . . . . .	66
4.4.2.6	Min_score_print_mean . . . . .	67
4.4.2.7	Min_verifValue . . . . .	68
4.5	metaboflux/include/gsl_mod.h File Reference . . . . .	68
4.5.1	Detailed Description . . . . .	69
4.5.2	Function Documentation . . . . .	70
4.5.2.1	Mod_compute_modeling . . . . .	70
4.5.2.2	Mod_score_print_mean . . . . .	71
4.6	metaboflux/include/gsl_recuit.h File Reference . . . . .	72
4.6.1	Detailed Description . . . . .	73
4.6.2	Function Documentation . . . . .	74
4.6.2.1	Recuit_compute_recuit . . . . .	74
4.6.2.2	Recuit_defParametre . . . . .	75
4.6.2.3	Recuit_energyFunction . . . . .	76
4.6.2.4	Recuit_metricDistance . . . . .	77
4.6.2.5	Recuit_printParametre . . . . .	78
4.6.2.6	Recuit_printPosition . . . . .	79
4.6.2.7	Recuit_redirectionFlux . . . . .	80
4.6.2.8	Recuit_takeStep . . . . .	81
4.6.2.9	Recuit_verifParameters . . . . .	81
4.6.2.10	Recuit_verifParameters_2 . . . . .	82
4.7	metaboflux/include/gsl_sd.h File Reference . . . . .	82
4.7.1	Detailed Description . . . . .	83
4.7.2	Function Documentation . . . . .	83
4.7.2.1	Sd_compute_simulation . . . . .	83
4.7.2.2	Sd_compute_standard_deviation . . . . .	85
4.8	metaboflux/include/mpi_load.h File Reference . . . . .	86
4.8.1	Detailed Description . . . . .	87
4.8.2	Function Documentation . . . . .	87
4.8.2.1	compute_mpi . . . . .	87
4.8.2.2	Mpi_allocResultTab . . . . .	90

4.8.2.3	Mpi_connectInterface . . . . .	90
4.8.2.4	Mpi_disconnectInterface . . . . .	90
4.8.2.5	Mpi_finalize . . . . .	91
4.8.2.6	Mpi_init . . . . .	91
4.8.2.7	Mpi_master . . . . .	91
4.8.2.8	Mpi_sizeResultTab . . . . .	92
4.8.2.9	Mpi_slave . . . . .	93
4.8.2.10	Mpi_writer . . . . .	95
4.8.2.11	Mpi_writeSdFile . . . . .	95
4.8.2.12	Mpi_writeSimFile . . . . .	96
4.9	metabolflux/include/simulation.h File Reference . . . . .	96
4.9.1	Detailed Description . . . . .	98
4.9.2	Function Documentation . . . . .	98
4.9.2.1	SBML_allocTest . . . . .	98
4.9.2.2	SBML_checkQuantite . . . . .	99
4.9.2.3	SBML_compute_simulation . . . . .	99
4.9.2.4	SBML_compute_simulation_mean . . . . .	101
4.9.2.5	SBML_debugPrint . . . . .	102
4.9.2.6	SBML_debugPrintHead . . . . .	103
4.9.2.7	SBML_EstimationReaction . . . . .	103
4.9.2.8	SBML_evalExpression . . . . .	104
4.9.2.9	SBML_findReaction . . . . .	104
4.9.2.10	SBML_freeTest . . . . .	105
4.9.2.11	SBML_initEspeceAmounts . . . . .	105
4.9.2.12	SBML_reactChoice . . . . .	106
4.9.2.13	SBML_reaction . . . . .	106
4.9.2.14	SBML_score . . . . .	107
4.9.2.15	SBML_score_add . . . . .	108
4.9.2.16	SBML_score_mean . . . . .	109
4.9.2.17	SBML_setReactions . . . . .	109
4.9.2.18	SBML_simulate . . . . .	110
4.10	metabolflux/include/xml_parameter.h File Reference . . . . .	111
4.10.1	Detailed Description . . . . .	113
4.10.2	Function Documentation . . . . .	114

---

4.10.2.1	Xml_allocEquation . . . . .	114
4.10.2.2	Xml_freeConfig . . . . .	114
4.10.2.3	Xml_getallSpeciesFinalAmount . . . . .	115
4.10.2.4	Xml_getallSpeciesWeight . . . . .	115
4.10.2.5	Xml_getBanned . . . . .	116
4.10.2.6	Xml_getBoltzmann . . . . .	117
4.10.2.7	Xml_getCount . . . . .	117
4.10.2.8	Xml_getDoubleNumber . . . . .	118
4.10.2.9	Xml_getKineticLaw . . . . .	118
4.10.2.10	Xml_getKineticLawNodes . . . . .	119
4.10.2.11	Xml_getMuT . . . . .	120
4.10.2.12	Xml_getNbBannedSpecies . . . . .	120
4.10.2.13	Xml_getNbCouples . . . . .	121
4.10.2.14	Xml_getNbEquations . . . . .	122
4.10.2.15	Xml_getNbGroup . . . . .	122
4.10.2.16	Xml_getNbIter . . . . .	123
4.10.2.17	Xml_getNbParameters . . . . .	124
4.10.2.18	Xml_getNbReaction . . . . .	124
4.10.2.19	Xml_getNbReactioninNoeud . . . . .	125
4.10.2.20	Xml_getNbSimulations . . . . .	126
4.10.2.21	Xml_getNbSpecies . . . . .	126
4.10.2.22	Xml_getNbTriesMod . . . . .	127
4.10.2.23	Xml_getNbTriesSa . . . . .	128
4.10.2.24	Xml_getNumber . . . . .	128
4.10.2.25	Xml_getReactionsNames . . . . .	129
4.10.2.26	Xml_getReactionsNamesinNoeud . . . . .	129
4.10.2.27	Xml_getSpeciesFinalAmount . . . . .	130
4.10.2.28	Xml_getSpeciesWeight . . . . .	131
4.10.2.29	Xml_getStepSize . . . . .	131
4.10.2.30	Xml_getString . . . . .	132
4.10.2.31	Xml_getTinitial . . . . .	132
4.10.2.32	Xml_getTmin . . . . .	133
4.10.2.33	Xml_loadConfig . . . . .	134
4.11	metaboflux/src/data_parameters.c File Reference . . . . .	134

4.11.1	Detailed Description	136
4.11.2	Function Documentation	137
4.11.2.1	Data_allocEquations	137
4.11.2.2	Data_allocParameters	137
4.11.2.3	Data_allocScore	138
4.11.2.4	Data_allocSimParameters	139
4.11.2.5	Data_copieTab	139
4.11.2.6	Data_desallocation	140
4.11.2.7	Data_desallocSim	141
4.11.2.8	Data_equationsAlloc	141
4.11.2.9	Data_equationsInit	141
4.11.2.10	Data_initSBML	143
4.11.2.11	Data_initXML	144
4.11.2.12	Data_parameters	144
4.11.2.13	Data_rngAlloc	145
4.11.2.14	Data_scoreAlloc	146
4.11.2.15	Data_scoreFree	146
4.11.2.16	Data_scoreInit	147
4.11.2.17	Data_simParameters	147
4.11.2.18	Data_updateTab	148
4.12	metaboflux/src/equations.c File Reference	149
4.12.1	Detailed Description	151
4.12.2	Function Documentation	151
4.12.2.1	add	151
4.12.2.2	divide	152
4.12.2.3	equality	152
4.12.2.4	Equations_addOp	152
4.12.2.5	Equations_alloc	153
4.12.2.6	Equations_calcul	154
4.12.2.7	Equations_defiler	155
4.12.2.8	Equations_defilerSuv	156
4.12.2.9	Equations_define	156
4.12.2.10	Equations_depiler	157
4.12.2.11	Equations_empiler	158

4.12.2.12 Equations_emptyOp . . . . .	158
4.12.2.13 Equations_extractData . . . . .	159
4.12.2.14 Equations_finalQuantite . . . . .	160
4.12.2.15 Equations_findSpecies . . . . .	160
4.12.2.16 Equations_pileFormation . . . . .	161
4.12.2.17 Equations_print . . . . .	163
4.12.2.18 Equations_priorite . . . . .	163
4.12.2.19 Equations_resultat . . . . .	163
4.12.2.20 Equations_sommet . . . . .	164
4.12.2.21 Equations_vide . . . . .	165
4.12.2.22 inf . . . . .	165
4.12.2.23 inf_equal . . . . .	165
4.12.2.24 multiply . . . . .	166
4.12.2.25 subtract . . . . .	166
4.12.2.26 sup . . . . .	166
4.12.2.27 sup_equal . . . . .	167
4.13 metaboflux/src/especies.c File Reference . . . . .	167
4.13.1 Detailed Description . . . . .	169
4.13.2 Function Documentation . . . . .	169
4.13.2.1 Especies_alloc . . . . .	169
4.13.2.2 Especies_allocReactions . . . . .	170
4.13.2.3 Especies_find . . . . .	170
4.13.2.4 Especies_free . . . . .	170
4.13.2.5 Especies_freeReactions . . . . .	171
4.13.2.6 Especies_getNbReactions . . . . .	171
4.13.2.7 Especies_getQuantite . . . . .	172
4.13.2.8 Especies_print . . . . .	172
4.13.2.9 Especies_print_2 . . . . .	172
4.13.2.10 Especies_save . . . . .	173
4.13.2.11 Especies_scoreSpecies . . . . .	173
4.13.2.12 Especies_setQuantite . . . . .	174
4.14 metaboflux/src/gsl_min.c File Reference . . . . .	174
4.14.1 Detailed Description . . . . .	175
4.14.2 Function Documentation . . . . .	176

---

4.14.2.1	Min_compute_minimization . . . . .	176
4.14.2.2	Min_copieTab2 . . . . .	177
4.14.2.3	Min_copieTab3 . . . . .	178
4.14.2.4	Min_getTampon . . . . .	178
4.14.2.5	Min_my_f . . . . .	179
4.14.2.6	Min_score_print_mean . . . . .	180
4.14.2.7	Min_verifValue . . . . .	181
4.15	metaboflux/src/gsl_mod.c File Reference . . . . .	181
4.15.1	Detailed Description . . . . .	182
4.15.2	Function Documentation . . . . .	183
4.15.2.1	Mod_compute_modeling . . . . .	183
4.15.2.2	Mod_score_print_mean . . . . .	184
4.16	metaboflux/src/gsl_recuit.c File Reference . . . . .	185
4.16.1	Detailed Description . . . . .	187
4.16.2	Function Documentation . . . . .	187
4.16.2.1	Recuit_compute_recuit . . . . .	187
4.16.2.2	Recuit_defParametre . . . . .	188
4.16.2.3	Recuit_energyFunction . . . . .	189
4.16.2.4	Recuit_metricDistance . . . . .	190
4.16.2.5	Recuit_printParametre . . . . .	191
4.16.2.6	Recuit_printPosition . . . . .	192
4.16.2.7	Recuit_redirectionFlux . . . . .	193
4.16.2.8	Recuit_takeStep . . . . .	194
4.16.2.9	Recuit_verifParameters . . . . .	194
4.16.2.10	Recuit_verifParameters_2 . . . . .	195
4.17	metaboflux/src/gsl_sd.c File Reference . . . . .	195
4.17.1	Detailed Description . . . . .	196
4.17.2	Function Documentation . . . . .	197
4.17.2.1	Sd_compute_simulation . . . . .	197
4.17.2.2	Sd_compute_standard_deviation . . . . .	198
4.18	metaboflux/src/MetaBoFlux.c File Reference . . . . .	199
4.18.1	Detailed Description . . . . .	201
4.18.2	Function Documentation . . . . .	202
4.18.2.1	alloc_arguments . . . . .	202

4.18.2.2	argument_analysis . . . . .	202
4.18.2.3	check_arguments . . . . .	203
4.18.2.4	check_common . . . . .	203
4.18.2.5	error_activity . . . . .	204
4.18.2.6	free_arguments . . . . .	204
4.18.2.7	help_print . . . . .	205
4.18.2.8	main . . . . .	205
4.19	metaboflux/src/mpi_load.c File Reference . . . . .	206
4.19.1	Detailed Description . . . . .	208
4.19.2	Function Documentation . . . . .	209
4.19.2.1	compute_mpi . . . . .	209
4.19.2.2	Mpi_allocResultTab . . . . .	211
4.19.2.3	Mpi_connectInterface . . . . .	211
4.19.2.4	Mpi_disconnectInterface . . . . .	211
4.19.2.5	Mpi_finalize . . . . .	212
4.19.2.6	Mpi_init . . . . .	212
4.19.2.7	Mpi_master . . . . .	212
4.19.2.8	Mpi_sizeResultTab . . . . .	213
4.19.2.9	Mpi_slave . . . . .	214
4.19.2.10	Mpi_writer . . . . .	216
4.19.2.11	Mpi_writeSdFile . . . . .	216
4.19.2.12	Mpi_writeSimFile . . . . .	217
4.20	metaboflux/src/simulation.c File Reference . . . . .	217
4.20.1	Detailed Description . . . . .	219
4.20.2	Function Documentation . . . . .	220
4.20.2.1	SBML_allocTest . . . . .	220
4.20.2.2	SBML_checkQuantite . . . . .	220
4.20.2.3	SBML_compute_simulation . . . . .	221
4.20.2.4	SBML_compute_simulation_mean . . . . .	222
4.20.2.5	SBML_debugPrint . . . . .	223
4.20.2.6	SBML_debugPrintHead . . . . .	224
4.20.2.7	SBML_EstimationReaction . . . . .	224
4.20.2.8	SBML_evalExpression . . . . .	225
4.20.2.9	SBML_findReaction . . . . .	225

4.20.2.10 SBML_freeTest . . . . .	226
4.20.2.11 SBML_initEspeceAmounts . . . . .	226
4.20.2.12 SBML_reactChoice . . . . .	227
4.20.2.13 SBML_reaction . . . . .	227
4.20.2.14 SBML_score . . . . .	228
4.20.2.15 SBML_score_add . . . . .	229
4.20.2.16 SBML_score_mean . . . . .	230
4.20.2.17 SBML_setReactions . . . . .	230
4.20.2.18 SBML_simulate . . . . .	231
4.21 metaboflux/src/xml_parameter.c File Reference . . . . .	232
4.21.1 Detailed Description . . . . .	235
4.21.2 Function Documentation . . . . .	235
4.21.2.1 Xml_allocEquation . . . . .	235
4.21.2.2 Xml_freeConfig . . . . .	236
4.21.2.3 Xml_getallSpeciesFinalAmount . . . . .	236
4.21.2.4 Xml_getallSpeciesWeight . . . . .	237
4.21.2.5 Xml_getBanned . . . . .	237
4.21.2.6 Xml_getBoltzmann . . . . .	238
4.21.2.7 Xml_getCount . . . . .	239
4.21.2.8 Xml_getDoubleNumber . . . . .	239
4.21.2.9 Xml_getKineticLaw . . . . .	239
4.21.2.10 Xml_getKineticLawNodes . . . . .	240
4.21.2.11 Xml_getMuT . . . . .	241
4.21.2.12 Xml_getNbBannedSpecies . . . . .	241
4.21.2.13 Xml_getNbCouples . . . . .	242
4.21.2.14 Xml_getNbEquations . . . . .	243
4.21.2.15 Xml_getNbGroup . . . . .	243
4.21.2.16 Xml_getNbIter . . . . .	244
4.21.2.17 Xml_getNbParameters . . . . .	245
4.21.2.18 Xml_getNbReaction . . . . .	245
4.21.2.19 Xml_getNbReactioninNoeud . . . . .	246
4.21.2.20 Xml_getNbSimulations . . . . .	247
4.21.2.21 Xml_getNbSpecies . . . . .	247
4.21.2.22 Xml_getNbTriesMod . . . . .	248

4.21.2.23 Xml_getNbTriesSa . . . . .	249
4.21.2.24 Xml_getNumber . . . . .	249
4.21.2.25 Xml_getReactionsNames . . . . .	250
4.21.2.26 Xml_getReactionsNamesinNoeud . . . . .	251
4.21.2.27 Xml_getSpeciesFinalAmount . . . . .	251
4.21.2.28 Xml_getSpeciesWeight . . . . .	252
4.21.2.29 Xml_getStepSize . . . . .	252
4.21.2.30 Xml_getString . . . . .	253
4.21.2.31 Xml_getTinitial . . . . .	253
4.21.2.32 Xml_getTmin . . . . .	254
4.21.2.33 Xml_loadConfig . . . . .	255

# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<code>Equations</code> (Function pointers used for operations ) . . . . .	5
<code>Equdata</code> (Equation data ) . . . . .	6
<code>Especies</code> (Molecules data ) . . . . .	7
<code>ListArguments</code> (List of arguments ) . . . . .	8
<code>ListParameters</code> (Global list parameters ) . . . . .	10
<code>option</code> (List of flag for getopt ) . . . . .	13
<code>Reaction</code> (Reactions data ) . . . . .	14
<code>Score</code> (Structure containing all information from simulations ) . . . . .	15
<code>SimParameters</code> (List of parameters specific to each simulation ) . . . . .	17
<code>TestReaction</code> (Structure used to test reactions ) . . . . .	19
<code>xmlConfig_t</code> (Structure containing all information from parameter.xml ) . . . . .	20



# Chapter 2

## File Index

### 2.1 File List

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

metaboflux/include/ <a href="#">data_parameters.h</a> (Load data parameters ) . . . . .	23
metaboflux/include/ <a href="#">equations.h</a> (Processes an equation in MathML format ) . . .	37
metaboflux/include/ <a href="#">especies.h</a> (Modelize a molecule ) . . . . .	55
metaboflux/include/ <a href="#">gsl_min.h</a> (Compute minimization of scenarii ) . . . . .	62
metaboflux/include/ <a href="#">gsl_mod.h</a> (Compute the modeling of scenarii ) . . . . .	68
metaboflux/include/ <a href="#">gsl_recuit.h</a> (Compute the simulated annealing ) . . . . .	72
metaboflux/include/ <a href="#">gsl_sd.h</a> (Compute the standard deviation analysis of the simulations ) . . . . .	82
metaboflux/include/ <a href="#">mpi_load.h</a> (Parallelize the program ) . . . . .	86
metaboflux/include/ <a href="#">simulation.h</a> (Simulate a petri net ) . . . . .	96
metaboflux/include/ <a href="#">xml_parameter.h</a> (Xml reader for parametre.xml ) . . . . .	111
metaboflux/src/ <a href="#">data_parameters.c</a> (Load data parameters ) . . . . .	134
metaboflux/src/ <a href="#">equations.c</a> (Processes an equation in MathML format ) . . . .	149
metaboflux/src/ <a href="#">especies.c</a> (Modelize a molecule ) . . . . .	167
metaboflux/src/ <a href="#">gsl_min.c</a> (Compute minimization of scenarii ) . . . . .	174
metaboflux/src/ <a href="#">gsl_mod.c</a> (Compute the modeling of scenarii ) . . . . .	181
metaboflux/src/ <a href="#">gsl_recuit.c</a> (Compute the simulated annealing ) . . . . .	185
metaboflux/src/ <a href="#">gsl_sd.c</a> (Compute the standard deviation analysis of the simulations ) . . . . .	195
metaboflux/src/ <a href="#">MetaBoFlux.c</a> (Program for simulated annealing analysis ) . . .	199
metaboflux/src/ <a href="#">mpi_load.c</a> (Parallelize the program ) . . . . .	206
metaboflux/src/ <a href="#">simulation.c</a> (Simulate a petri net ) . . . . .	217
metaboflux/src/ <a href="#">xml_parameter.c</a> (Xml reader for parametre.xml ) . . . . .	232



# Chapter 3

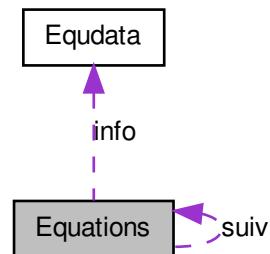
## Data Structure Documentation

### 3.1 Equations Struct Reference

Function pointers used for operations.

```
#include <equations.h>
```

Collaboration diagram for Equations:



#### Data Fields

- `Equtype type`
- `Equedata info`

##### 3.1.1 Detailed Description

Function pointers used for operations.

Definition at line 68 of file equations.h.

### 3.1.2 Field Documentation

#### 3.1.2.1 Equdata Equations::info

Equation Data

Definition at line 71 of file equations.h.

Referenced by Equations\_print().

#### 3.1.2.2 Equtype Equations::type

Operation constants

Definition at line 70 of file equations.h.

Referenced by Equations\_print(), and Equations\_resultat().

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

- metaboflux/include/equations.h

## 3.2 Equdata Union Reference

Equation data.

```
#include <equations.h>
```

### Data Fields

- double `data`
- char \* `var`

#### 3.2.1 Detailed Description

Equation data.

Definition at line 57 of file equations.h.

### 3.2.2 Field Documentation

#### 3.2.2.1 double Equdata::data

Constant

Definition at line 60 of file equations.h.

Referenced by Equations\_print().

### 3.2.2.2 char\* Equedata::var

Variable

Definition at line 61 of file equations.h.

Referenced by Equations\_print().

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

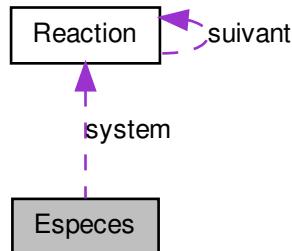
- metaboflux/include/[equations.h](#)

## 3.3 Espèces Struct Reference

Molecules data.

```
#include <especes.h>
```

Collaboration diagram for Espèces:



### Data Fields

- double [quantite](#)
- const char \* [id](#)
- [pReaction](#) system

### 3.3.1 Detailed Description

Molecules data.

Definition at line 45 of file especes.h.

### 3.3.2 Field Documentation

#### 3.3.2.1 const char\* Especies::id

Molecule id

Definition at line 47 of file especes.h.

Referenced by Especies\_save().

#### 3.3.2.2 double Especies::quantite

Molecule quantity

Definition at line 46 of file especes.h.

Referenced by Especies\_alloc(), Especies\_getQuantite(), Especies\_save(), Especies\_scoreSpecies(), and Especies\_setQuantite().

#### 3.3.2.3 pReaction Especies::system

[Reaction](#) where the molecule is implicated

Definition at line 48 of file especes.h.

Referenced by Especies\_alloc(), Especies\_allocReactions(), Especies\_freeReactions(), Especies\_getNbReactions(), Especies\_print(), SBML\_EstimationReaction(), SBML\_reactChoice(), and SBML\_simulate().

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

- metaboflux/include/[especies.h](#)

## 3.4 ListArguments Struct Reference

List of arguments.

### Data Fields

- int [activity](#)
- int [debug](#)
- int [port](#)

- int [group](#)
- char \*\* [files\\_path](#)

### 3.4.1 Detailed Description

List of arguments.

Definition at line 65 of file MetaBoFlux.c.

### 3.4.2 Field Documentation

#### 3.4.2.1 int ListArguments::activity

Activity chosen

Definition at line 67 of file MetaBoFlux.c.

Referenced by `alloc_arguments()`, `argument_analysis()`, `check_arguments()`, and `main()`.

#### 3.4.2.2 int ListArguments::debug

Debug chosen

Definition at line 68 of file MetaBoFlux.c.

Referenced by `alloc_arguments()`, `argument_analysis()`, and `main()`.

#### 3.4.2.3 char\*\* ListArguments::files\_path

Path to files path : 0.SBML 1.PAR 2.OUT 3.RATIO\_FILE 4.LOG 5.OUTPUT

Definition at line 71 of file MetaBoFlux.c.

Referenced by `alloc_arguments()`, `argument_analysis()`, `check_arguments()`, `check_common()`, `free_arguments()`, and `main()`.

#### 3.4.2.4 int ListArguments::group

Group chosen

Definition at line 70 of file MetaBoFlux.c.

Referenced by `alloc_arguments()`, `argument_analysis()`, and `main()`.

#### 3.4.2.5 int ListArguments::port

Port chosen

Definition at line 69 of file MetaBoFlux.c.

Referenced by alloc\_arguments(), argument\_analysis(), and main().

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

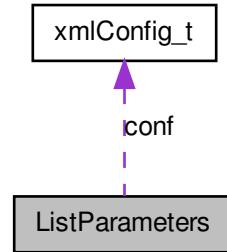
- metaboflux/src/[MetaBoFlux.c](#)

### 3.5 ListParameters Struct Reference

Global list parameters.

```
#include <data_parameters.h>
```

Collaboration diagram for ListParameters:



#### Data Fields

- Model\_t \* [model](#)
- [xmlConfig\\_t](#) \* [conf](#)
- char \*\*\*\* [equation](#)
- int [nb\\_parameters](#)
- int [interest\\_parameters](#)
- int [nb\\_couples](#)
- int [nb\\_equations](#)
- int [nb\\_reaction](#)
- int \* [noeud](#)
- int \* [parameters](#)
- char \*\* [banned](#)
- int [nb\\_banned](#)
- int [nb\\_triesMod](#)
- int [nb\\_triesSa](#)

### 3.5.1 Detailed Description

Global list parameters.

Definition at line 35 of file data\_parameters.h.

### 3.5.2 Field Documentation

#### 3.5.2.1 `char** ListParameters::banned`

Names of banned molecules : ex ATP, ADP ...

Definition at line 47 of file data\_parameters.h.

Referenced by Data\_allocParameters(), Data\_desallocation(), Min\_my\_f(), Mod\_compute\_-modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

#### 3.5.2.2 `xmlConfig_t* ListParameters::conf`

Parameter file

Definition at line 38 of file data\_parameters.h.

Referenced by Data\_allocEquations(), Data\_allocParameters(), Data\_desallocation(), Data\_parameters(), Data\_scoreAlloc(), Data\_simParameters(), Min\_my\_f(), Mod\_compute\_-modeling(), Mpi\_master(), Mpi\_sizeResultTab(), Mpi\_slave(), Mpi\_writer(), Recuit\_compute\_-recuit(), Recuit\_printParametre(), and Sd\_compute\_standard\_deviation().

#### 3.5.2.3 `char**** ListParameters::equation`

Equations text

Definition at line 39 of file data\_parameters.h.

Referenced by Data\_allocEquations(), Data\_desallocation(), and Data\_equationsInit().

#### 3.5.2.4 `int ListParameters::interest_parameters`

Number of interest parameters

Definition at line 41 of file data\_parameters.h.

Referenced by Data\_allocParameters(), Min\_compute\_minimization(), Recuit\_compute\_-recuit(), Recuit\_metricDistance(), Recuit\_printParametre(), and Recuit\_takeStep().

#### 3.5.2.5 `Model_t* ListParameters::model`

SBML file

Definition at line 37 of file data\_parameters.h.

Referenced by Data\_desallocation(), Data\_parameters(), Data\_scoreAlloc(), Data\_simParameters(), Min\_my\_f(), Mod\_compute\_modeling(), Mpi\_sizeResultTab(), Mpi\_writer(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

### 3.5.2.6 int ListParameters::nb\_banned

Number of banned molecules

Definition at line 48 of file data\_parameters.h.

Referenced by Data\_allocParameters(), Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

### 3.5.2.7 int ListParameters::nb\_couples

Number of couples

Definition at line 42 of file data\_parameters.h.

Referenced by Data\_allocParameters(), Data\_updateTab(), Min\_compute\_minimization(), Min\_my\_f(), Recuit\_defParametre(), and Recuit\_takeStep().

### 3.5.2.8 int ListParameters::nb\_equations

Number of equations

Definition at line 43 of file data\_parameters.h.

Referenced by Data\_allocEquations(), Data\_desallocation(), Data\_equationsAlloc(), Data\_equationsInit(), Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

### 3.5.2.9 int ListParameters::nb\_parameters

Number of parameters

Definition at line 40 of file data\_parameters.h.

Referenced by Data\_allocParameters(), Data\_allocSimParameters(), Data\_simParameters(), Min\_compute\_minimization(), Min\_getTampon(), Min\_my\_f(), Min\_score\_print\_mean(), Mod\_compute\_modeling(), Mod\_score\_print\_mean(), Mpi\_master(), Mpi\_sizeResultTab(), Mpi\_slave(), Mpi\_writeSimFile(), Recuit\_printPosition(), and Sd\_compute\_standard\_deviation().

### 3.5.2.10 int ListParameters::nb\_reaction

Number of reaction

Definition at line 44 of file data\_parameters.h.

**3.5.2.11 int ListParameters::nb\_triesMod**

Number of tries for modelling

Definition at line 49 of file data\_parameters.h.

Referenced by Data\_allocParameters().

**3.5.2.12 int ListParameters::nb\_triesSa**

Number of tries for simulated annealing and minimization

Definition at line 50 of file data\_parameters.h.

Referenced by Data\_allocParameters(), and Recuit\_energyFunction().

**3.5.2.13 int\* ListParameters::noeud**

Number of nodes in an equation

Definition at line 45 of file data\_parameters.h.

Referenced by Data\_allocEquations(), Data\_desallocation(), and Data\_equationsInit().

**3.5.2.14 int\* ListParameters::parameters**

Details on the parameters of reactions

Definition at line 46 of file data\_parameters.h.

Referenced by Data\_allocParameters(), Data\_desallocation(), Data\_updateTab(), Min\_compute\_minimization(), Min\_my\_f(), Recuit\_defParametre(), and Recuit\_takeStep().

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

- metaboflux/include/[data\\_parameters.h](#)

## 3.6 option Struct Reference

List of flag for getopt.

### 3.6.1 Detailed Description

List of flag for getopt.

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

- metaboflux/src/[MetaBoFlux.c](#)

## 3.7 Reaction Struct Reference

Reactions data.

```
#include <especies.h>
```

Collaboration diagram for Reaction:



### Data Fields

- Reaction\_t \* [link](#)
- struct [Reaction](#) \* [suivant](#)

#### 3.7.1 Detailed Description

Reactions data.

Definition at line 34 of file especes.h.

#### 3.7.2 Field Documentation

##### 3.7.2.1 Reaction\_t\* Reaction::link

[Reaction](#) id

Definition at line 36 of file especes.h.

Referenced by Especies\_allocReactions(), Especies\_print(), SBML\_EstimationReaction(), SBML\_reactChoice(), and SBML\_simulate().

##### 3.7.2.2 struct Reaction\* Reaction::suivant

Next reaction implicated for the molecule

Definition at line 38 of file especes.h.

Referenced by Especies\_allocReactions(), Especies\_freeReactions(), Especies\_getNbReactions(), Especies\_print(), SBML\_EstimationReaction(), and SBML\_reactChoice().

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

- metaboflux/include/especies.h

## 3.8 Score Struct Reference

Structure containing all information from simulations.

```
#include <simulation.h>
```

### Data Fields

- char \*\* name
- double \* quantite
- char \*\* reaction
- int nb\_reaction
- int tailleReactions
- int tailleSpecies
- int taille
- char \*\* species
- int nb\_species
- int \* species\_amount
- int \* species\_weight

### 3.8.1 Detailed Description

Structure containing all information from simulations.

Definition at line 51 of file simulation.h.

### 3.8.2 Field Documentation

#### 3.8.2.1 char\*\* Score::name

Molecules id

Definition at line 53 of file simulation.h.

Referenced by Data\_allocScore(), Data\_desallocSim(), Data\_scoreFree(), Min\_my\_f(), Min\_score\_print\_mean(), Mod\_compute\_modeling(), Mod\_score\_print\_mean(), Recuit\_-energyFunction(), SBML\_score(), SBML\_score\_add(), and Sd\_compute\_simulation().

#### 3.8.2.2 int Score::nb\_reaction

Number of reaction

Definition at line 56 of file simulation.h.

Referenced by Data\_allocScore(), Data\_desallocSim(), Data\_scoreFree(), SBML\_score(), and SBML\_setReactions().

### 3.8.2.3 int Score::nb\_species

Number of molecules

Definition at line 61 of file simulation.h.

Referenced by Data\_allocScore(), Min\_my\_f(), Min\_score\_print\_mean(), Mod\_compute\_-modeling(), Mod\_score\_print\_mean(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

### 3.8.2.4 double\* Score::quantite

Quantity of molecules

Definition at line 54 of file simulation.h.

Referenced by Data\_allocScore(), Data\_desallocSim(), Data\_scoreFree(), Data\_scoreInit(), Min\_getTampon(), Min\_my\_f(), Min\_score\_print\_mean(), Mod\_compute\_modeling(), Mod\_score\_print\_mean(), Recuit\_energyFunction(), SBML\_score(), SBML\_score\_add(), SBML\_score\_mean(), and Sd\_compute\_simulation().

### 3.8.2.5 char\*\* Score::reaction

Interest reaction

Definition at line 55 of file simulation.h.

Referenced by Data\_allocScore(), Data\_desallocSim(), Data\_scoreFree(), SBML\_score(), and SBML\_setReactions().

### 3.8.2.6 char\*\* Score::species

Interest molecules

Definition at line 60 of file simulation.h.

Referenced by Data\_allocScore(), Data\_desallocSim(), Min\_my\_f(), Min\_score\_print\_mean(), Mod\_compute\_modeling(), Mod\_score\_print\_mean(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

### 3.8.2.7 int\* Score::species\_amount

Quantity of the interest molecules

Definition at line 62 of file simulation.h.

Referenced by Data\_allocScore(), Data\_desallocSim(), Data\_scoreFree(), Min\_my\_f(), Min\_score\_print\_mean(), Mod\_compute\_modeling(), Mod\_score\_print\_mean(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

### 3.8.2.8 int\* Score::species\_weight

Weight of the interest molecules

Definition at line 63 of file simulation.h.

Referenced by Data\_allocScore(), Data\_desallocSim(), Data\_scoreFree(), Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

### 3.8.2.9 int Score::taille

Total of other two sizes

Definition at line 59 of file simulation.h.

Referenced by Data\_allocScore(), Data\_desallocSim(), Data\_scoreFree(), Data\_scoreInit(), Min\_getTampon(), Min\_my\_f(), Min\_score\_print\_mean(), Mod\_compute\_modeling(), Mod\_score\_print\_mean(), Recuit\_energyFunction(), SBML\_score\_add(), SBML\_score\_mean(), and Sd\_compute\_simulation().

### 3.8.2.10 int Score::tailleReactions

Number of reaction in the SBML file

Definition at line 57 of file simulation.h.

Referenced by Data\_allocScore(), and SBML\_compute\_simulation().

### 3.8.2.11 int Score::tailleSpecies

Number of molecules in the SBML file

Definition at line 58 of file simulation.h.

Referenced by Data\_allocScore(), Min\_my\_f(), Min\_score\_print\_mean(), Mod\_compute\_modeling(), Mod\_score\_print\_mean(), Recuit\_energyFunction(), SBML\_compute\_simulation(), SBML\_score\_add(), and Sd\_compute\_simulation().

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

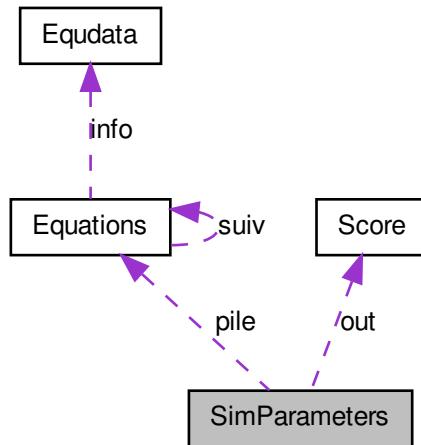
- metaboflux/include/[simulation.h](#)

## 3.9 SimParameters Struct Reference

List of parameters specific to each simulation.

```
#include <data_parameters.h>
```

Collaboration diagram for SimParameters:



## Data Fields

- `Equations ** pile`
- `pScore out`
- `gsl_rng * r`
- `double * y`
- `FILE * debugFile`

### 3.9.1 Detailed Description

List of parameters specific to each simulation.

Definition at line 58 of file `data_parameters.h`.

### 3.9.2 Field Documentation

#### 3.9.2.1 FILE\* SimParameters::debugFile

Debug file : storage of intermediate values of the simulations

Definition at line 64 of file `data_parameters.h`.

Referenced by `Data_desallocSim()`, `Data_simParameters()`, `Min_my_f()`, `Mod_compute_modeling()`, `Recuit_energyFunction()`, and `Sd_compute_simulation()`.

**3.9.2.2 pScore SimParameters::out****Score**

Definition at line 61 of file data\_parameters.h.

Referenced by Data\_desallocSim(), Data\_simParameters(), Min\_getTampon(), Min\_my\_f(), Min\_score\_print\_mean(), Mod\_compute\_modeling(), Mod\_score\_print\_mean(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

**3.9.2.3 Equations\*\* SimParameters::pile****Interpreted Equations**

Definition at line 60 of file data\_parameters.h.

Referenced by Data\_simParameters(), Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

**3.9.2.4 gsl\_rng\* SimParameters::r**

Random number generator

Definition at line 62 of file data\_parameters.h.

Referenced by Data\_desallocSim(), Data\_simParameters(), Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_compute\_recuit(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

**3.9.2.5 double\* SimParameters::y**

Table of reaction parameters

Definition at line 63 of file data\_parameters.h.

Referenced by Data\_copieTab(), Data\_desallocSim(), Data\_simParameters(), Min\_compute\_minimization(), Min\_copieTab2(), Min\_copieTab3(), Min\_getTampon(), Min\_my\_f(), Min\_score\_print\_mean(), Mod\_compute\_modeling(), Mod\_score\_print\_mean(), Recuit\_energyFunction(), Recuit\_printPosition(), Sd\_compute\_simulation(), and Sd\_compute\_standard\_deviation().

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

- metaboflux/include/[data\\_parameters.h](#)

## 3.10 TestReaction Struct Reference

Structure used to test reactions.

```
#include <simulation.h>
```

## Data Fields

- Reaction\_t \*\* [tabReactions](#)
- int \* [minStepTab](#)

### 3.10.1 Detailed Description

Structure used to test reactions.

Definition at line 40 of file simulation.h.

### 3.10.2 Field Documentation

#### 3.10.2.1 int\* [TestReaction::minStepTab](#)

Stock le nombre de realisation possible pour cette reaction

Definition at line 43 of file simulation.h.

Referenced by SBML\_allocTest(), SBML\_EstimationReaction(), and SBML\_freeTest().

#### 3.10.2.2 Reaction\_t\*\* [TestReaction::tabReactions](#)

Stock temporairement une reaction

Definition at line 42 of file simulation.h.

Referenced by SBML\_allocTest(), SBML\_EstimationReaction(), and SBML\_freeTest().

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

- metaboflux/include/[simulation.h](#)

## 3.11 xmlConfig\_t Struct Reference

Structure containing all information from parameter.xml.

```
#include <xml_parameter.h>
```

## Data Fields

- char \* [fichier](#)
- xmlDocPtr [doc](#)
- xmlNodePtr [racine](#)
- xmlXPathContextPtr [ctxt](#)

### 3.11.1 Detailed Description

Structure containing all information from parameter.xml.

Definition at line 33 of file xml\_parameter.h.

### 3.11.2 Field Documentation

#### 3.11.2.1 `xmlXPathContextPtr xmlConfig_t::ctxt`

Xml ctxt

Definition at line 37 of file xml\_parameter.h.

Referenced by `Xml_freeConfig()`, `Xml_getBanned()`, `Xml_getCount()`, `Xml_getDoubleNumber()`, `Xml_getKineticLaw()`, `Xml_getKineticLawNodes()`, `Xml_getNumber()`, `Xml_getReactionsNames()`, `Xml_getReactionsNamesinNoeud()`, `Xml_getSpeciesFinalAmount()`, `Xml_getSpeciesWeight()`, `XmlGetString()`, and `Xml_loadConfig()`.

#### 3.11.2.2 `xmlDocPtr xmlConfig_t::doc`

Xml doc

Definition at line 35 of file xml\_parameter.h.

Referenced by `Xml_freeConfig()`, and `Xml_loadConfig()`.

#### 3.11.2.3 `char* xmlConfig_t::fichier`

Name of the parameter file

Definition at line 34 of file xml\_parameter.h.

Referenced by `Xml_freeConfig()`, and `Xml_loadConfig()`.

#### 3.11.2.4 `xmlNodePtr xmlConfig_t::racine`

Xml root

Definition at line 36 of file xml\_parameter.h.

Referenced by `Xml_loadConfig()`.

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

- metaboflux/include/[xml\\_parameter.h](#)



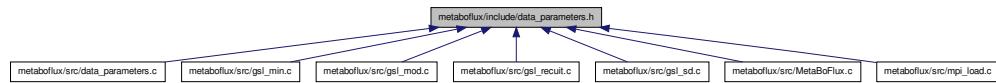
## Chapter 4

# File Documentation

### 4.1 metaboflux/include/data\_parameters.h File Reference

Load data parameters.

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



#### Data Structures

- struct [ListParameters](#)  
*Global list parameters.*
- struct [SimParameters](#)  
*List of parameters specific to each simulation.*

#### Functions

- [xmlConfig\\_t \\* Data\\_initXML \(char \\*\)](#)  
*Load parameter file.*
- [Model\\_t \\* Data\\_initSBML \(char \\*\)](#)  
*Load SBML file.*
- [gsl\\_rng \\* Data\\_rngAlloc \(gsl\\_rng \\*\)](#)  
*Allocation of the random number generator.*
- [Equations \\*\\*\\* Data\\_equationsAlloc \(pListParameters, Equations \\*\\*\\*\)](#)  
*Allocate the struct [Equations](#).*

- void **Data\_equationsInit** (pListParameters, Equations \*\*)  
*Initialize equations.*
- void **Data\_allocEquations** (pListParameters a)  
*Allocate the different elements related to the struct Equations.*
- void **Data\_allocScore** (pScore, xmlConfig\_t \*, Model\_t \*)  
*Copy the partial table x in the full table y.*
- pScore **Data\_scoreAlloc** (pListParameters)  
*Allocation of the struct Score.*
- void **Data\_scoreInit** (pScore)  
*Initialize the struct Score.*
- void **Data\_scoreFree** (pScore)  
*Free the struct Score.*
- int **Data\_copieTab** (pSimParameters, double \*, int, int, int)  
*Copy the partial table x in the full table y.*
- void **Data\_updateTab** (pListParameters, pSimParameters, double \*)  
*Copy table.*
- void **Data\_allocSimParameters** (double \*, pListParameters)  
*Allocation of tables needed for simulations.*
- void **Data\_allocParameters** (pListParameters)  
*Copy the partial table x in the full table y.*
- void **Data\_parameters** (pListParameters, char \*\*)  
*Record parameters of the simulation.*
- void **Data\_desallocation** (pListParameters)  
*Free memory allocated to the various objects of the program.*
- void **Data\_simParameters** (pListParameters, pSimParameters, char \*, char \*, int, int)  
*Recording parameters specific to each simulation.*
- void **Data\_desallocSim** (pSimParameters)  
*Free struct parameters specific to each simulation.*

#### 4.1.1 Detailed Description

Load data parameters. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metabofl>)  
 Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

{Amine Ghozlane}

**Version**

2.0

**Date**

10 novembre 2010

Definition in file [data\\_parameters.h](#).

## 4.1.2 Function Documentation

### 4.1.2.1 void Data\_allocEquations ( pListParameters a )

Allocate the different elements related to the struct [Equations](#).

**Author**

Amine Ghozlane

**Parameters**

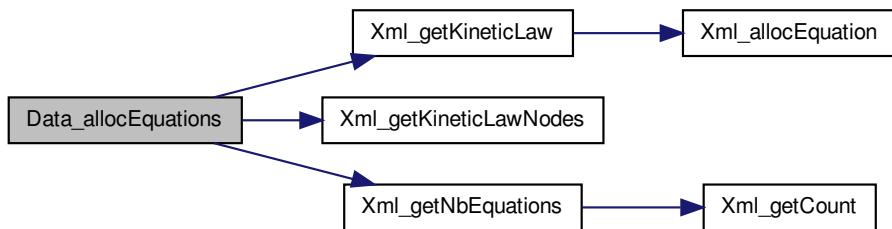
a	Global parameters : struct <a href="#">ListParameters</a>
---	---

Definition at line 202 of file [data\\_parameters.c](#).

References [ListParameters::conf](#), [ListParameters::equation](#), [ListParameters::nb\\_equations](#), [ListParameters::noeud](#), [Xml\\_getKineticLaw\(\)](#), [Xml\\_getKineticLawNodes\(\)](#), and [Xml\\_getNbEquations\(\)](#).

Referenced by [Data\\_parameters\(\)](#).

Here is the call graph for this function:



#### 4.1.2.2 void Data\_allocParameters ( pListParameters a )

Copy the partial table x in the full table y.

##### Author

Amine Ghozlane

##### Parameters

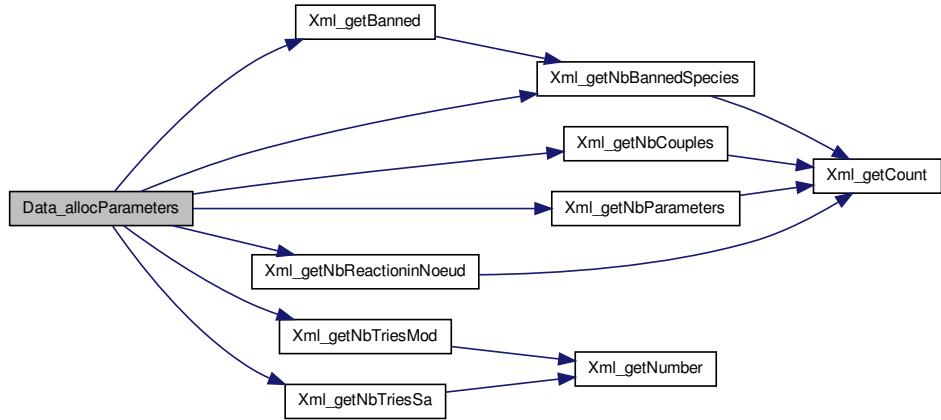
a	Global parameters : struct <a href="#">ListParameters</a>
---	---

Definition at line 408 of file `data_parameters.c`.

References `ListParameters::banned`, `ListParameters::conf`, `ListParameters::interest_parameters`, `ListParameters::nb_banned`, `ListParameters::nb_couples`, `ListParameters::nb_parameters`, `ListParameters::nb_triesMod`, `ListParameters::nb_triesSa`, `ListParameters::parameters`, `Xml_getBanned()`, `Xml_getNbBannedSpecies()`, `Xml_getNbCouples()`, `Xml_getNbParameters()`, `Xml_getNbReactioninNoeud()`, `Xml_getNbTriesMod()`, and `Xml_getNbTriesSa()`.

Referenced by `Data_parameters()`.

Here is the call graph for this function:



#### 4.1.2.3 void Data\_allocScore ( pScore out, xmlConfig\_t \* conf, Model\_t \* model )

Copy the partial table x in the full table y.

##### Author

Amine Ghozlane

**Parameters**

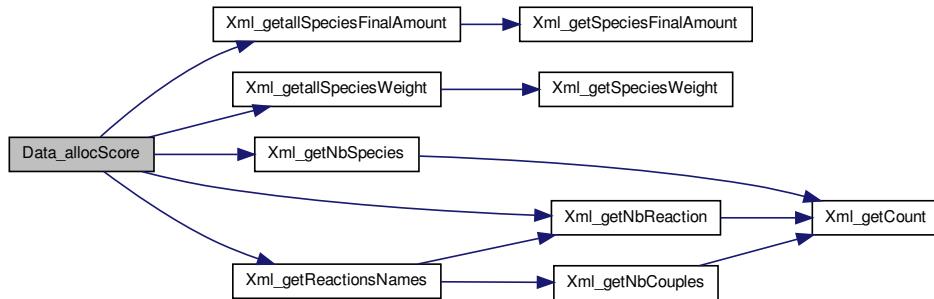
<i>out</i>	Struct <a href="#">Score</a>
<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>model</i>	Model of the SBML file

Definition at line 238 of file data\_parameters.c.

References Score::name, Score::nb\_reaction, Score::nb\_species, Score::quantite, Score::reaction, Score::species, Score::species\_amount, Score::species\_weight, Score::taille, Score::tailleReactions, Score::tailleSpecies, Xml\_getallSpeciesFinalAmount(), Xml\_getallSpeciesWeight(), Xml\_getNbReaction(), Xml\_getNbSpecies(), and Xml\_getReactionsNames().

Referenced by Data\_scoreAlloc(), and Data\_simParameters().

Here is the call graph for this function:

**4.1.2.4 void Data\_allocSimParameters ( double \* y, pListParameters a )**

Allocation of tables needed for simulations.

[void Data\\_allocSimParameters\(double \\*y, pListParameters a\)](#)

**Author**

Amine Ghozlane

**Parameters**

<i>y</i>	table of reaction parameters
<i>a</i>	Global parameters : struct <a href="#">ListParameters</a>

Definition at line 391 of file data\_parameters.c.

References ListParameters::nb\_parameters.

Referenced by Data\_simParameters().

---

**4.1.2.5 int Data\_copieTab ( pSimParameters *simulated*, double \* *x*, int *a*, int *debut*, int *fin* )**

Copy the partial table x in the full table y.

**Author**

Amine Ghozlane

**Parameters**

<i>simulated</i>	Simulation parameters : struct <a href="#">SimParameters</a>
<i>x</i>	Short table of reaction parameters
<i>a</i>	Line
<i>debut</i>	Beginning
<i>fin</i>	End

**Returns**

Number of copied element

Definition at line 347 of file `data_parameters.c`.

References `SimParameters::y`.

Referenced by `Data_updateTab()`.

**4.1.2.6 void Data\_desallocation ( pListParameters *a* )**

Free memory allocated to the various objects of the program.

**Author**

Amine Ghozlane

**Parameters**

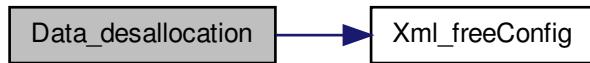
<i>a</i>	Global parameters : struct <a href="#">ListParameters</a>
----------	---

Definition at line 463 of file `data_parameters.c`.

References `ListParameters::banned`, `ListParameters::conf`, `ListParameters::equation`, `ListParameters::model`, `ListParameters::nb_equations`, `ListParameters::noeud`, `ListParameters::parameters`, and `Xml_freeConfig()`.

Referenced by `compute_mpi()`.

Here is the call graph for this function:



#### 4.1.2.7 void Data\_desallocSim ( pSimParameters sim )

Free struct parameters specific to each simulation.

##### Author

Amine Ghozlane

##### Parameters

<i>sim</i>	Simulation parameters : struct <a href="#">SimParameters</a>
------------	--

Definition at line 549 of file data\_parameters.c.

References SimParameters::debugFile, Score::name, Score::nb\_reaction, SimParameters::out, Score::quantite, SimParameters::r, Score::reaction, Score::species, Score::species\_amount, Score::species\_weight, Score::taille, and SimParameters::y.

Referenced by Min\_compute\_minimization(), Mod\_compute\_modeling(), Recuit\_compute\_recuit(), and Sd\_compute\_standard\_deviation().

#### 4.1.2.8 Equations\*\* Data\_equationsAlloc ( pListParameters current, Equations \*\* pile )

Allocate the struct [Equations](#).

##### Author

Amine Ghozlane

##### Parameters

<i>current</i>	Current parameters : struct <a href="#">ListParameters</a>
<i>pile</i>	Pile des equations : struct <a href="#">Equations</a>

##### Returns

Allocated struct [Equations](#)

Definition at line 169 of file data\_parameters.c.

References ListParameters::nb\_equations.

Referenced by Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

#### 4.1.2.9 void Data\_equationsInit ( pListParameters *current*, Equations \*\* *pile* )

Initialize equations.

##### Author

Amine Ghozlane

##### Parameters

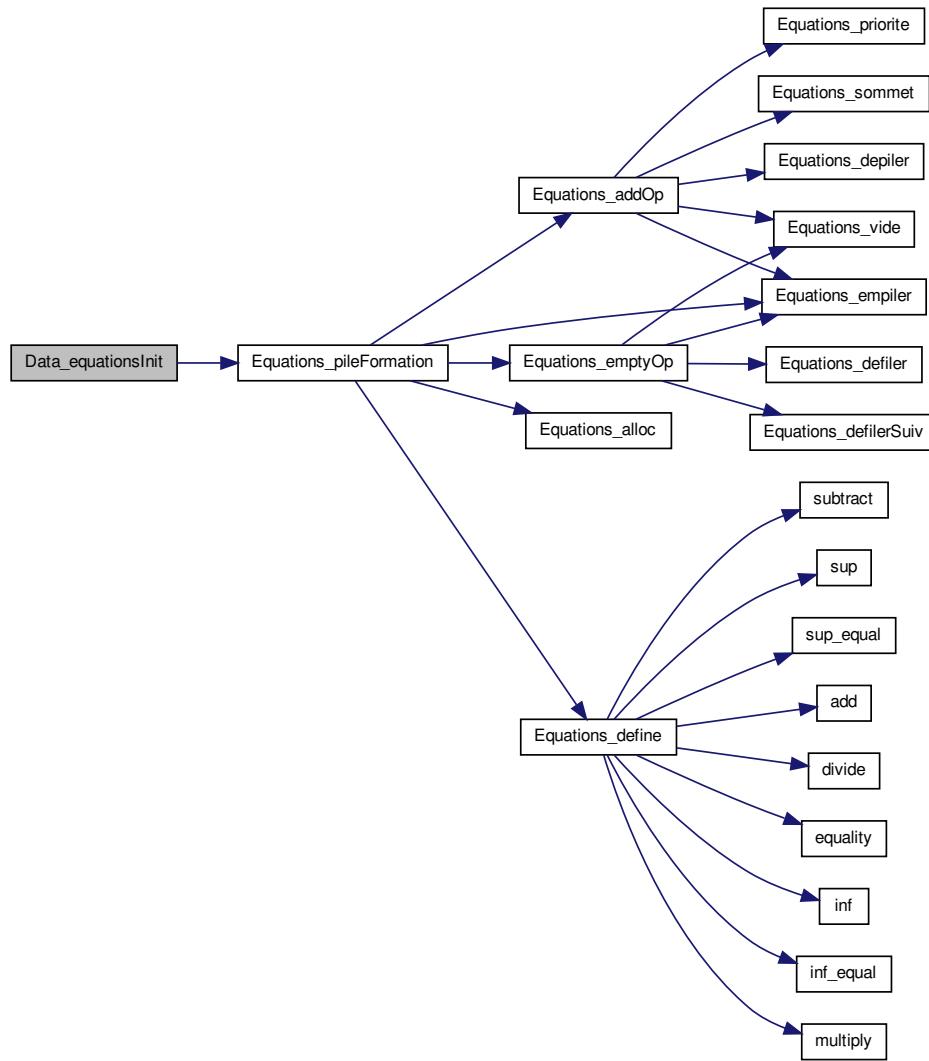
<i>current</i>	Current parameters : struct <a href="#">ListParameters</a>
<i>pile</i>	Pile des equations : struct <a href="#">Equations</a>

Definition at line 184 of file data\_parameters.c.

References ListParameters::equation, Equations\_pileFormation(), ListParameters::nb\_equations, and ListParameters::noeud.

Referenced by Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

Here is the call graph for this function:



#### 4.1.2.10 Model\_t\* Data\_initSBML ( char \* sbml\_file )

Load SBML file.

## Author

Amine Ghozlane

**Parameters**

<i>sbml_file</i>	Name of the SBML file
------------------	-----------------------

**Returns**

Model of the SBML file

Definition at line 81 of file data\_parameters.c.

Referenced by Data\_parameters().

**4.1.2.11 *xmlConfig\_t\** Data\_initXML ( *char \* xml\_file* )**

Load parameter file.

**Author**

Amine Ghozlane

**Parameters**

<i>xml_file</i>	Name of the parameter file
-----------------	----------------------------

**Returns**

Model of the parameter file =:Struct [xmlConfig\\_t](#)

Definition at line 58 of file data\_parameters.c.

References Xml\_loadConfig().

Referenced by Data\_parameters().

Here is the call graph for this function:

**4.1.2.12 void Data\_parameters ( *pListParameters a, char \*\* files\_path* )**

Record parameters of the simulation.

**Author**

Amine Ghozlane

**Parameters**

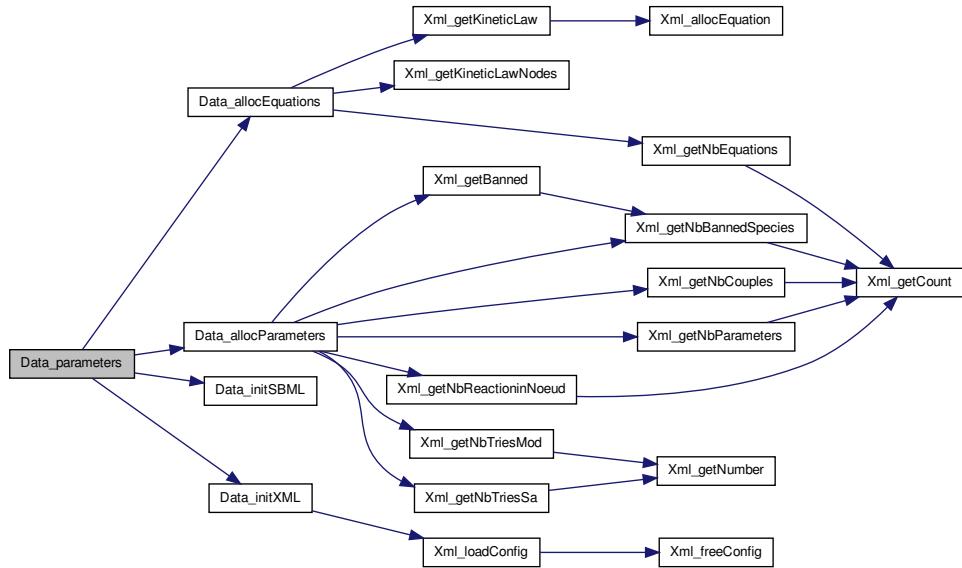
<i>a</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>files_path</i>	List of paths

Definition at line 442 of file data\_parameters.c.

References ListParameters::conf, Data\_allocEquations(), Data\_allocParameters(), Data\_initSBML(), Data\_initXML(), and ListParameters::model.

Referenced by compute\_mpi().

Here is the call graph for this function:

**4.1.2.13 gsl\_rng\* Data\_rngAlloc ( gsl\_rng \* r )**

Allocation of the random number generator.

**Author**

Amine Ghozlane

**Parameters**

<i>r</i>	Random number generator
----------	-------------------------

**Returns**

Random number generator

Definition at line 135 of file data\_parameters.c.

Referenced by Data\_simParameters().

#### 4.1.2.14 pScore Data\_scoreAlloc ( pListParameters a )

Allocation of the struct [Score](#).

##### Author

Amine Ghozlane

##### Parameters

<i>a</i>	Global parameters : struct <a href="#">ListParameters</a>
----------	---

##### Returns

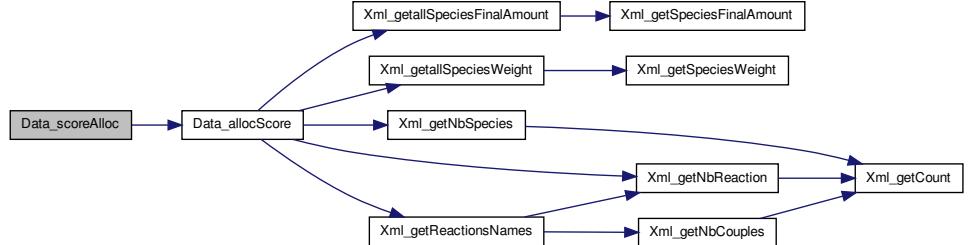
Allocated struct [Score](#)

Definition at line 274 of file data\_parameters.c.

References ListParameters::conf, Data\_allocScore(), and ListParameters::model.

Referenced by Min\_compute\_minimization(), Mod\_compute\_modeling(), and Recuit\_compute\_recuit().

Here is the call graph for this function:



#### 4.1.2.15 void Data\_scoreFree ( pScore out )

Free the struct [Score](#).

##### Author

Amine Ghozlane

##### Parameters

<i>out</i>	Struct score
------------	--------------

Definition at line 307 of file data\_parameters.c.

References Score::name, Score::nb\_reaction, Score::quantite, Score::reaction, Score::species\_amount, Score::species\_weight, and Score::taille.

Referenced by Min\_compute\_minimization(), Mod\_compute\_modeling(), and Recuit\_compute\_recuit().

#### 4.1.2.16 void Data\_scoreInit ( pScore *out* )

Initialize the struct [Score](#).

##### Author

Amine Ghozlane

##### Parameters

<i>out</i>	Empty struct <a href="#">Score</a>
------------	------------------------------------

Definition at line 290 of file data\_parameters.c.

References Score::quantite, and Score::taille.

Referenced by Min\_my\_f(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

#### 4.1.2.17 void Data\_simParameters ( pListParameters *a*, pSimParameters *sim*, char \* *out*, char \* *texte*, int *number*, int *debug* )

Recording parameters specific to each simulation.

##### Author

Amine Ghozlane

##### Parameters

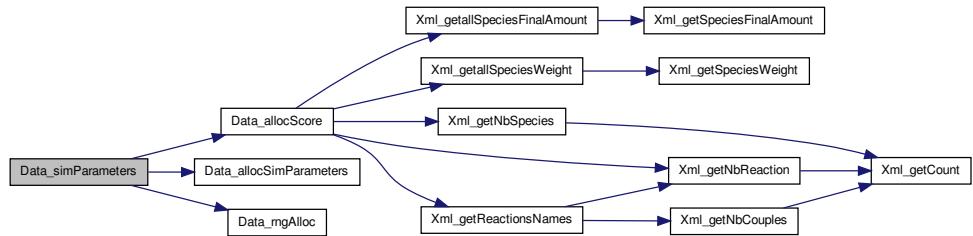
<i>a</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>sim</i>	Simulation parameters : struct <a href="#">SimParameters</a>
<i>out</i>	Output repertory
<i>texte</i>	File name
<i>number</i>	Number of the debug file
<i>debug</i>	Determine debug

Definition at line 506 of file data\_parameters.c.

References ListParameters::conf, Data\_allocScore(), Data\_allocSimParameters(), Data\_rngAlloc(), SimParameters::debugFile, ListParameters::model, ListParameters::nb\_parameters, SimParameters::out, SimParameters::pile, SimParameters::r, and SimParameters::y.

Referenced by Min\_compute\_minimization(), Mod\_compute\_modeling(), Recuit\_compute\_recuit(), and Sd\_compute\_standard\_deviation().

Here is the call graph for this function:



**4.1.2.18 void Data\_updateTab ( pListParameters *current*, pSimParameters *simulated*, double \* *x* )**

Copy table.

#### Author

Amine Ghozlane

#### Parameters

<i>current</i>	Current parameters : struct <a href="#">ListParameters</a>
<i>simulated</i>	Simulation parameters : struct <a href="#">SimParameters</a>
<i>x</i>	Short table of reaction parameters

Definition at line 371 of file `data_parameters.c`.

References `Data_copieTab()`, `ListParameters::nb_couples`, and `ListParameters::parameters`.

Referenced by `Min_my_f()`, `Recuit_energyFunction()`, and `Recuit_printPosition()`.

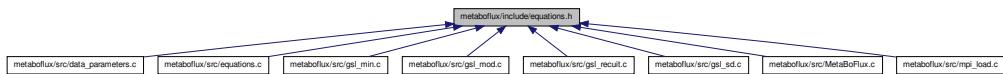
Here is the call graph for this function:



## 4.2 metaboflux/include/equations.h File Reference

Processes an equation in MathML format.

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



### Data Structures

- union [Equdata](#)  
*Equation data.*
- struct [Equations](#)  
*Function pointers used for operations.*

### Enumerations

- enum [Equtype](#) {  
    constant, variable, addition, soustraction,  
    multiplication, division, equal, superior,  
    superior\_equal, inferior, inferior\_equal }  
*Operation constants.*

### Functions

- double [add](#) (double, double)  
*Add two numbers.*
- double [subtract](#) (double, double)  
*Subtraction two numbers.*
- double [multiply](#) (double, double)  
*Multiply two numbers.*
- double [divide](#) (double, double)  
*Divide two numbers.*
- double [equality](#) (double, double)  
*Test the equality of two numbers.*
- double [sup](#) (double, double)  
*Test the superiority between two numbers.*
- double [sup\\_equal](#) (double, double)  
*Test the superiority or equality between two numbers.*

- double `inf` (double, double)  
*Test the inferiority between two numbers.*
- double `inf_equal` (double, double)  
*Test the inferiority or equality between two numbers.*
- `pEquations Equations_alloc` (void)  
*Alloc memory and initialize the struct `Equations`.*
- void `Equations_define` (`pEquations`, char \*)  
*Identify the mathematical operator used.*
- int `Equations_vide` (`pEquations`)  
*Test if the struct `Equations` is empty.*
- `pEquations Equations_empiler` (`pEquations`, `pEquations`)  
*Stack an element to the struct `Equations`.*
- `pEquations Equations_depiler` (`pEquations`)  
*Unstack the last element of the struct `Equations`.*
- `pEquations Equations_sommet` (`pEquations`)  
*Look for the last element of the struct `Equations`.*
- `pEquations Equations_defiler` (`pEquations`)  
*Look for the first element of the struct `Equations`.*
- `pEquations Equations_defilerSuiv` (`pEquations`)  
*Get the next element of the struct `Equations`.*
- int `Equations_priorite` (`pEquations`)  
*Give the priority of the operator. Multiplication and Division have higher priority than addition or subtraction...*
- void `Equations_print` (`pEquations`)  
*Print all information on the element of the struct `Equations`.*
- `pEquations Equations_addOp` (`pEquations`, `pEquations`, `pEquations`)  
*Build the list of the Struct `Equations` result used to compute the equation.*
- void `Equations_emptyOp` (`pEquations`, `pEquations`)  
*Empty the Struct `Equations` op for the struct Equation result.*
- `pEquations Equations_pileFormation` (char \*\*\*, int)  
*Build pile.*
- double `Equations_findSpecies` (char \*, char \*\*, double \*, int)  
*Look for the quantity of a molecule in the list.*
- double `Equations_resultat` (`pEquations`, `pEquations`, `pEquations`, char \*\*, double \*, int)  
*Compute the result of two pile.*
- double `Equations_calcul` (`pEquations`, char \*\*, double \*, int)  
*Compute the score of the equation.*
- double `Equations_finalQuantite` (int, char \*\*, int \*, int \*, char \*\*, double \*, int)  
*Compute the score of the quantity which means the difference between what is expected to what is simulated.*

### 4.2.1 Detailed Description

Processes an equation in MathML format. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/meta>)  
Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

{Amine Ghozlane}

#### Version

2.0

#### Date

15 janvier 2010

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

### 4.2.2 Enumeration Type Documentation

#### 4.2.2.1 enum Equtype

Operation constants.

Equtype is a set of predefined constants for the different operators.

##### Enumerator:

```
constant Constant
variable Variable : ex. transition_x ...
addition Addition
soustraction Subtraction
multiplication Multiply
division Divide
equal Equality
superior Superiority
superior_equal Sup_equality
inferior Inferiority
inferior_equal Inf_equality
```

Definition at line 38 of file equations.h.

---

### 4.2.3 Function Documentation

#### 4.2.3.1 double add ( double *arg1*, double *arg2* )

Add two numbers.

##### Author

Amine Ghozlane

##### Parameters

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

##### Returns

Result

Definition at line 43 of file equations.c.

Referenced by Equations\_define().

#### 4.2.3.2 double divide ( double *arg1*, double *arg2* )

Divide two numbers.

##### Author

Amine Ghozlane

##### Parameters

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

##### Returns

Result

Definition at line 82 of file equations.c.

Referenced by Equations\_define().

#### 4.2.3.3 double equality ( double *arg1*, double *arg2* )

Test the equality of two numbers.

##### Author

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 95 of file equations.c.

Referenced by Equations\_define().

**4.2.3.4 pEquations Equations\_addOp ( pEquations *result*, pEquations *op*, pEquations *new* )**Build the list of the Struct [Equations](#) result used to compute the equation.**Author**

Amine Ghozlane

**Parameters**

<i>result</i>	Struct <a href="#">Equations</a> used for computation
<i>op</i>	Struct <a href="#">Equations</a> used to store the operator
<i>new</i>	One element of the Struct <a href="#">Equations</a>

**Returns**

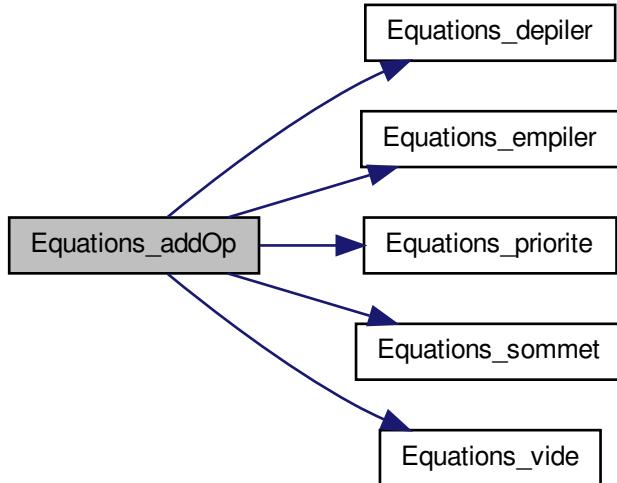
List of operator

Definition at line 383 of file equations.c.

References equal, Equations\_depiler(), Equations\_empiler(), Equations\_priorite(), Equations\_sommet(), Equations\_vide(), inferior, inferior\_equal, superior, and superior\_equal.

Referenced by Equations\_pileFormation().

Here is the call graph for this function:



#### 4.2.3.5 pEquations Equations\_alloc ( void )

Alloc memory and initialize the struct [Equations](#).

##### Author

Amine Ghozlane

##### Returns

Allocated struct [Equations](#)

Definition at line 158 of file equations.c.

Referenced by [Equations\\_calcul\(\)](#), and [Equations\\_pileFormation\(\)](#).

#### 4.2.3.6 double Equations\_calcul ( pEquations liste, char \*\* name, double \* quantite, int taille )

Compute the score of the equation.

##### Author

Amine Ghozlane

**Parameters**

<i>liste</i>	pile of type Struct Equations
<i>name</i>	List of molecules
<i>quantite</i>	List of the quantity of the molecules
<i>taille</i>	Number of molecules in the list

**Returns**

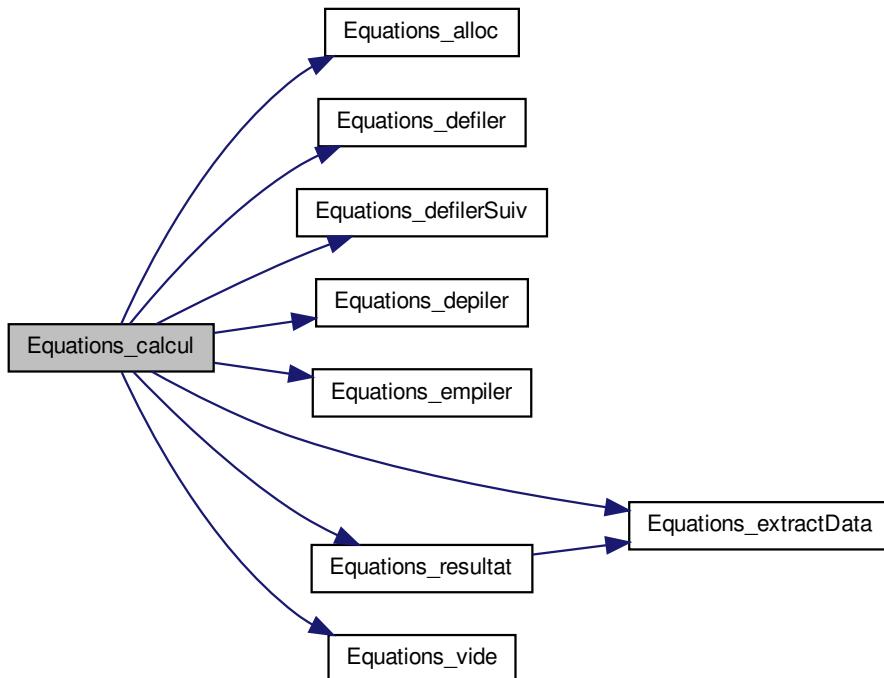
Result of the equation (First part of the energy)

Definition at line 614 of file equations.c.

References addition, constant, division, equal, Equations\_alloc(), Equations\_defiler(), Equations\_defilerSuiv(), Equations\_depiler(), Equations\_empiler(), Equations\_extractData(), Equations\_resultat(), Equations\_vide(), inferior, inferior\_equal, multiplication, soustraction, superior, and superior\_equal.

Referenced by Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

Here is the call graph for this function:



#### 4.2.3.7 pEquations Equations\_defiler ( pEquations liste )

Look for the first element of the struct [Equations](#).

##### Author

Amine Ghozlane

##### Parameters

<i>liste</i>	Struct <a href="#">Equations</a>
--------------	----------------------------------

##### Returns

Pointer on the first element of struct [Equations](#)

Definition at line 320 of file equations.c.

Referenced by [Equations\\_calcul\(\)](#), and [Equations\\_emptyOp\(\)](#).

#### 4.2.3.8 pEquations Equations\_defilerSuiv ( pEquations liste )

Get the next element of the struct [Equations](#).

##### Author

Amine Ghozlane

##### Parameters

<i>liste</i>	Struct <a href="#">Equations</a>
--------------	----------------------------------

##### Returns

Pointer on the next element of struct [Equations](#)

Definition at line 334 of file equations.c.

Referenced by [Equations\\_calcul\(\)](#), and [Equations\\_emptyOp\(\)](#).

#### 4.2.3.9 void Equations\_define ( pEquations new, char \* operateur )

Identify the mathematical operator used.

##### Author

Amine Ghozlane

##### Parameters

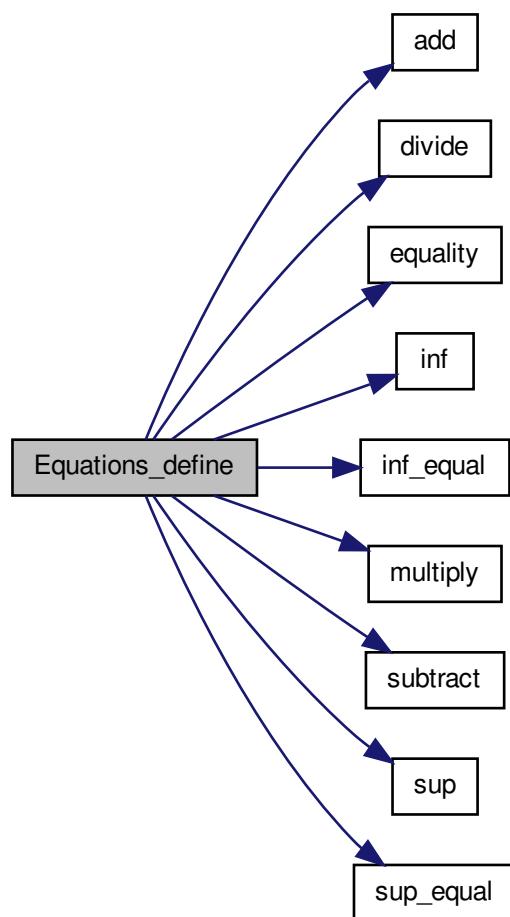
<i>new</i>	Struct <a href="#">Equations</a>
<i>operateur</i>	Read operator

Definition at line 174 of file equations.c.

References add(), addition, divide(), division, equal, equality(), inf(), inf\_equal(), inferior, inferior\_equal, multiplication, multiply(), soustraction, subtract(), sup(), sup\_equal(), superior, and superior\_equal.

Referenced by Equations\_pileFormation().

Here is the call graph for this function:



#### 4.2.3.10 pEquations Equations\_depiler ( pEquations liste )

Unstack the last element of the struct [Equations](#).

##### Author

Amine Ghozlane

##### Parameters

<i>liste</i>	Struct <a href="#">Equations</a> .
--------------	------------------------------------

##### Returns

Pointer on the "unstack" element of struct [Equations](#).

Definition at line 282 of file equations.c.

Referenced by [Equations\\_addOp\(\)](#), and [Equations\\_calcul\(\)](#).

#### 4.2.3.11 pEquations Equations\_empiler ( pEquations liste, pEquations new )

Stack an element to the struct [Equations](#).

##### Author

Amine Ghozlane

##### Parameters

<i>liste</i>	Struct <a href="#">Equations</a>
<i>new</i>	New struct <a href="#">Equations</a> element

##### Returns

Pointer on the last element of struct [Equations](#)

Definition at line 258 of file equations.c.

Referenced by [Equations\\_addOp\(\)](#), [Equations\\_calcul\(\)](#), [Equations\\_emptyOp\(\)](#), and [Equations\\_pileFormation\(\)](#).

#### 4.2.3.12 void Equations\_emptyOp ( pEquations result, pEquations op )

Empty the Struct [Equations](#) op for the struct Equation result.

##### Author

Amine Ghozlane

##### Parameters

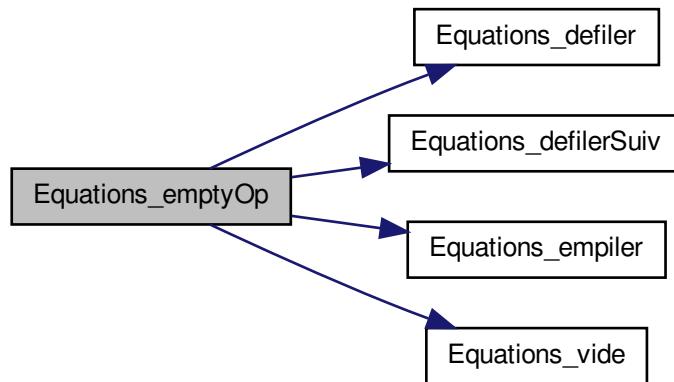
<i>result</i>	Struct <a href="#">Equations</a> used for computation
<i>op</i>	Struct Equation used to store the operator

Definition at line 421 of file equations.c.

References Equations\_defiler(), Equations\_defilerSuiv(), Equations\_empiler(), and Equations\_vide().

Referenced by Equations\_pileFormation().

Here is the call graph for this function:



**4.2.3.13** double Equations\_finalQuantite ( int *file\_nb\_especies*, char \*\* *file\_species*, int \* *file\_amount*, int \* *file\_weight*, char \*\* *sim\_name*, double \* *sim\_quantite*, int *sim\_taille* )

Compute the score of the quantity which means the difference between what is expected to what is simulated.

#### Author

Amine Ghozlane

#### Parameters

<i>file_nb_especies</i>	Number of species in the parameter file
<i>file_species</i>	List of species in the parameter file
<i>file_amount</i>	List of species expected quantity
<i>file_weight</i>	Weight defined for the species
<i>sim_name</i>	List of species simulated (sbml file)
<i>sim_quantite</i>	List of the quantity of the species simulated (sbml file)
<i>sim_taille</i>	Number of species simulated (sbml file)

**Returns**

Result of the difference (Second part of the Energy)

Definition at line 683 of file equations.c.

References Equations\_findSpecies().

Referenced by Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

Here is the call graph for this function:



**4.2.3.14 double Equations\_findSpecies ( char \* species, char \*\* name, double \* quantite, int taille )**

Look for the quantity of a molecule in the list.

**Author**

Amine Ghozlane

**Parameters**

<i>species</i>	Name of a molecule
<i>name</i>	List of molecules
<i>quantite</i>	List of the quantity of the molecules
<i>taille</i>	Number of molecules in the list

**Returns**

Quantity of the molecule of interest

Definition at line 533 of file equations.c.

Referenced by Equations\_finalQuantite(), Min\_score\_print\_mean(), and Mod\_score\_-print\_mean().

**4.2.3.15 pEquations Equations\_pileFormation ( char \*\*\* equation, int nb\_noeud )**

Build pile.

**Author**

Amine Ghozlane

**Parameters**

<i>equation</i>	Table with the equation in Mathml format
<i>nb_noeud</i>	Number of element inside the equation

**Returns**

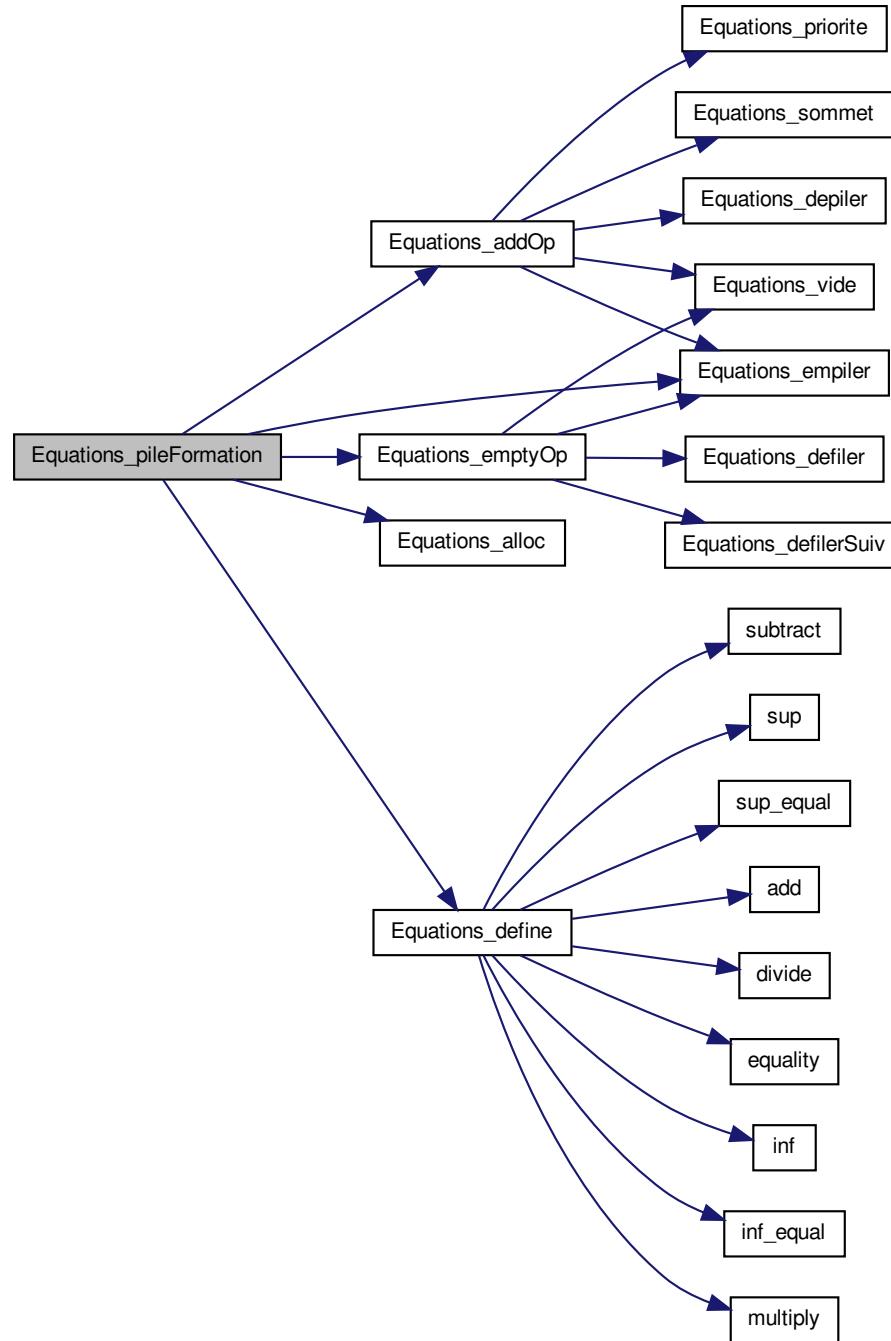
Struct Equation needed for compute the equation

Definition at line 441 of file equations.c.

References constant, Equations\_addOp(), Equations\_alloc(), Equations\_define(), Equations\_-empiler(), Equations\_emptyOp(), and variable.

Referenced by Data\_equationsInit().

Here is the call graph for this function:



**4.2.3.16 void Equations\_print ( pEquations liste )**

Print all information on the element of the struct [Equations](#).

**Author**

Amine Ghozlane

**Parameters**

<i>liste</i>	Struct <a href="#">Equations</a>
--------------	----------------------------------

Definition at line 359 of file equations.c.

References constant, Equdata::data, Equations::info, Equations::type, Equdata::var, and variable.

**4.2.3.17 int Equations\_priorite ( pEquations new )**

Give the priority of the operator. Multiplication and Division have higher priority than addition or subtraction...

**Author**

Amine Ghozlane

**Parameters**

<i>new</i>	One element of the struct <a href="#">Equations</a>
------------	---

**Returns**

Priority of the operator

Definition at line 346 of file equations.c.

References addition, equal, inferior, inferior\_equal, soustraction, superior, and superior\_-equal.

Referenced by Equations\_addOp().

**4.2.3.18 double Equations\_resultat ( pEquations result, pEquations result1, pEquations eq, char \*\* name, double \* quantite, int taille )**

Compute the result of two pile.

**Author**

Amine Ghozlane

**Parameters**

<i>result</i>	Struct Equation result
---------------	------------------------

<i>result1</i>	Struct Equation result after the operator
<i>eq</i>	Operator
<i>name</i>	List of molecules
<i>quantite</i>	List of the quantity of the molecules
<i>taille</i>	Number of molecules in the list

**Returns**

Result of the two pile

Definition at line 561 of file equations.c.

References equal, Equations\_extractData(), inferior, inferior\_equal, superior, superior\_-equal, and Equations::type.

Referenced by Equations\_calcul().

Here is the call graph for this function:

**4.2.3.19 pEquations Equations\_sommet ( pEquations liste )**

Look for the last element of the struct [Equations](#).

**Author**

Amine Ghozlane

**Parameters**

<i>liste</i>	Struct <a href="#">Equations</a>
--------------	----------------------------------

**Returns**

Pointer on the last element of struct [Equations](#)

Definition at line 305 of file equations.c.

Referenced by Equations\_addOp().

**4.2.3.20 int Equations\_vide ( pEquations liste )**

Test if the struct [Equations](#) is empty.

**Author**

Amine Ghozlane

**Parameters**

<i>liste</i>	Struct <a href="#">Equations</a>
--------------	----------------------------------

**Returns**

Number of elements

Definition at line 237 of file equations.c.

Referenced by [Equations\\_addOp\(\)](#), [Equations\\_calcul\(\)](#), and [Equations\\_emptyOp\(\)](#).

**4.2.3.21 double inf ( double arg1, double arg2 )**

Test the inferiority between two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 134 of file equations.c.

Referenced by [Equations\\_define\(\)](#).

**4.2.3.22 double inf\_equal ( double arg1, double arg2 )**

Test the inferiority or equality between two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 147 of file equations.c.

Referenced by Equations\_define().

**4.2.3.23 double multiply ( double *arg1*, double *arg2* )**

Multiply two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 69 of file equations.c.

Referenced by Equations\_define().

**4.2.3.24 double subtract ( double *arg1*, double *arg2* )**

Subtraction two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 56 of file equations.c.

Referenced by Equations\_define().

**4.2.3.25 double sup ( double *arg1*, double *arg2* )**

Test the superiority between two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 108 of file equations.c.

Referenced by Equations\_define().

**4.2.3.26 double sup\_equal ( double *arg1*, double *arg2* )**

Test the superiority or equality between two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

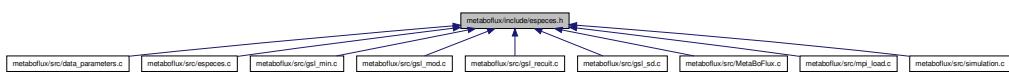
Definition at line 121 of file equations.c.

Referenced by Equations\_define().

**4.3 metaboflux/include/especies.h File Reference**

Modelize a molecule.

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



## Data Structures

- struct `Reaction`  
*Reactions data.*
- struct `Especies`  
*Molecules data.*

## Functions

- `pEspecies Especies_alloc (int)`  
*Alloc memory and initialize the struct `Especies`.*
- `void Especies_allocReactions (pEspecies, int, Reaction_t *, double)`  
*Alloc memory and initialize the struct `Reaction` for each molecule.*
- `void Especies_free (pEspecies, int)`  
*Free the struct `pEspecies`.*
- `void Especies_freeReactions (Especies)`  
*Free the struct `Reaction`.*
- `void Especies_save (pEspecies, int, double, const char *)`  
*Save data on one molecule.*
- `void Especies_scoreSpecies (pEspecies, int, char **, double *)`  
*Save the quantite of all molecules for scoring.*
- `void Especies_print (pEspecies, int)`  
*Print data on each molecule.*
- `void Especies_print_2 (pEspecies molecules, int nbEspecies)`  
*Print Id and quantity on each molecule.*
- `int Especies_find (pEspecies, const char *, int)`  
*Find a molecule thank to its ID.*
- `void Especies_setQuantite (pEspecies, int, double)`  
*Change the quantity of a molecule.*
- `double Especies_getQuantite (pEspecies, int)`  
*Get the quantity of one molecule by its number.*
- `int Especies_getNbReactions (pEspecies molecules, int ref)`  
*Get number of reaction where one molecule is used (like a reactif)*

### 4.3.1 Detailed Description

Modelize a molecule. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflux>)  
 Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

{Amine Ghozlane}

**Version**

2.0

**Date**

27 octobre 2009

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

### 4.3.2 Function Documentation

#### 4.3.2.1 `pEspecies` `Especies_alloc ( int nbEspecies )`

Alloc memory and initialize the struct [Especies](#).

**Author**

Amine Ghozlane

**Parameters**

<code>nbEspecies</code>	Number of molecules
-------------------------	---------------------

**Returns**

Allocated struct `Especie`

Definition at line 40 of file `especies.c`.

References `Especies::quantite`, and `Especies::system`.

Referenced by `SBML_compute_simulation()`, and `SBML_compute_simulation_mean()`.

#### 4.3.2.2 `void` `Especies_allocReactions ( pEspecies molecules, int Espece, Reaction_t * local, double reactRatio )`

Alloc memory and initialize the struct [Reaction](#) for each molecule.

**Author**

Amine Ghozlane

**Parameters**

<i>molecules</i>	List of molecules
<i>Espece</i>	Number of the selected molecule
<i>local</i>	Reaction sbml reference
<i>reactRatio</i>	Ratio of the reaction

Definition at line 64 of file especes.c.

References Reaction::link, Reaction::suivant, and Especies::system.

Referenced by SBML\_setReactions().

#### 4.3.2.3 int Especies\_find ( pEspecies *molecules*, const char \* *especeld*, int *nbEspecies* )

Find a molecule thank to it ID.

**Author**

Amine Ghozlane

**Parameters**

<i>molecules</i>	List of molecules
<i>especeld</i>	SBML ID of the molecule
<i>nbEspecies</i>	Number of molecules

**Returns**

Number of a selected molecule

Definition at line 214 of file especes.c.

Referenced by SBML\_checkQuantite(), SBML\_reaction(), and SBML\_setReactions().

#### 4.3.2.4 void Especies\_free ( pEspecies *molecules*, int *nbEspecies* )

Free the struct pEspecies.

**Author**

Amine Ghozlane

**Parameters**

<i>molecules</i>	List of molecules
<i>nbEspecies</i>	Number of molecules

Definition at line 94 of file especes.c.

References Especies\_freeReactions().

Referenced by SBML\_compute\_simulation(), and SBML\_compute\_simulation\_mean().

Here is the call graph for this function:



#### 4.3.2.5 void Especies\_freeReactions ( *Especies molecules* )

Free the struct Reaction.

##### Author

Amine Ghozlane

##### Parameters

<i>molecules</i>	List of molecules
------------------	-------------------

Definition at line 109 of file especes.c.

References Reaction::suivant, and Especies::system.

Referenced by Especies\_free().

#### 4.3.2.6 int Especies\_getNbReactions ( *pEspecies molecules*, int *ref* )

Get number of reaction where one molecule is used (like a reactif)

##### Author

Amine Ghozlane

##### Parameters

<i>molecules</i>	List of molecules
<i>ref</i>	Number of a selected molecule

##### Returns

Number of reactions

Definition at line 265 of file especes.c.

References Reaction::suivant, and Especies::system.

Referenced by SBML\_simulate().

#### 4.3.2.7 double Especies\_getQuantite ( pEspecies *molecules*, int *Especies* )

Get the quantity of one molecule by its number.

##### Author

Amine Ghozlane

##### Parameters

<i>molecules</i>	List of molecules
<i>Especies</i>	Number of a selected molecule

##### Returns

Quantity of a selected molecule

Definition at line 252 of file especes.c.

References Especies::quantite.

Referenced by SBML\_checkQuantite(), SBML\_reaction(), and SBML\_simulate().

#### 4.3.2.8 void Especies\_print ( pEspecies *molecules*, int *nbEspecies* )

Print data on each molecule.

##### Author

Amine Ghozlane

##### Parameters

<i>molecules</i>	List of molecules
<i>nbEspecies</i>	Number of molecules

Definition at line 170 of file especes.c.

References Reaction::link, Reaction::suivant, and Especies::system.

#### 4.3.2.9 void Especies\_print\_2 ( pEspecies *molecules*, int *nbEspecies* )

Print Id and quantity on each molecule.

##### Author

Amine Ghozlane

##### Parameters

---

<i>molecules</i>	List of molecules
<i>nbEspecies</i>	Number of molecules

Definition at line 196 of file especes.c.

4.3.2.10 void Especies\_save ( pEspecies *molecules*, int *i*, double *quant*, const char \* *dye* )

Save data on one molecule.

#### Author

Amine Ghozlane

#### Parameters

<i>molecules</i>	List of molecules
<i>i</i>	Number of the selected molecule
<i>quant</i>	Quantity of the molecule
<i>dye</i>	ID of the molecule

Definition at line 134 of file especes.c.

References Especies::id, and Especies::quantite.

Referenced by SBML\_initEspeceAmounts().

4.3.2.11 void Especies\_scoreSpecies ( pEspecies *molecules*, int *nbEspecies*, char \*\* *name*, double \* *quantite* )

Save the quantite of all molecules for scoring.

#### Author

Amine Ghozlane

#### Parameters

<i>molecules</i>	List of molecules
<i>nbEspecies</i>	Number of molecules
<i>name</i>	ID of the selected molecule
<i>quantite</i>	Table of molecule Quantity

Definition at line 149 of file especes.c.

References Especies::quantite.

Referenced by SBML\_score().

#### 4.3.2.12 void Especies\_setQuantite ( pEspecies molecules, int Especies, double quant )

Change the quantity of a molecule.

##### Author

Amine Ghozlane

##### Parameters

<i>molecules</i>	List of molecules
<i>Especies</i>	Number of a selected molecule
<i>quant</i>	Quantity of the molecule

Definition at line 239 of file especies.c.

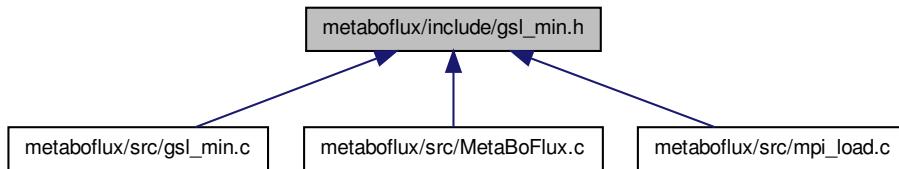
References Especies::quantite.

Referenced by SBML\_reaction().

## 4.4 metaboflux/include/gsl\_min.h File Reference

Compute minimization of scenarii.

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



## Functions

- int [Min\\_copieTab2](#) (double \*, int, int, int)  
*Copy table of reaction parameters.*
- int [Min\\_copieTab3](#) (gsl\_multimin\_fminimizer \*, int, int, int)  
*Copy short table of parameter into the larger table.*
- void [Min\\_verifValue](#) (gsl\_vector \*, double \*, int, int)  
*Checking reaction ratio parameters.*
- double [Min\\_my\\_f](#) (const gsl\_vector \*, void \*)  
*Enter in the program for standard deviation.*

- void [Min\\_getTampon](#) (double \*, double)  
*Enter in the program for standard deviation.*
- void [Min\\_score\\_print\\_mean](#) (double \*)  
*Save the minimization result.*
- void [Min\\_compute\\_minimization](#) (pListParameters, double \*, double \*, char \*\*, int, int)  
*Compute the minimization.*

#### 4.4.1 Detailed Description

Compute minimization of scenarii. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflu>)  
Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

##### Author

{Amine Ghozlane}

##### Version

2.0

##### Date

9 novembre 2009

Definition in file [gsl\\_min.h](#).

#### 4.4.2 Function Documentation

##### 4.4.2.1 void Min\_compute\_minimization ( pListParameters *alone*, double \* *fluxRatio*, double \* *result\_tab*, char \*\* *files\_path*, int *number\_arg*, int *debug* )

Compute the minimization.

##### Author

Amine Ghozlane

##### Parameters

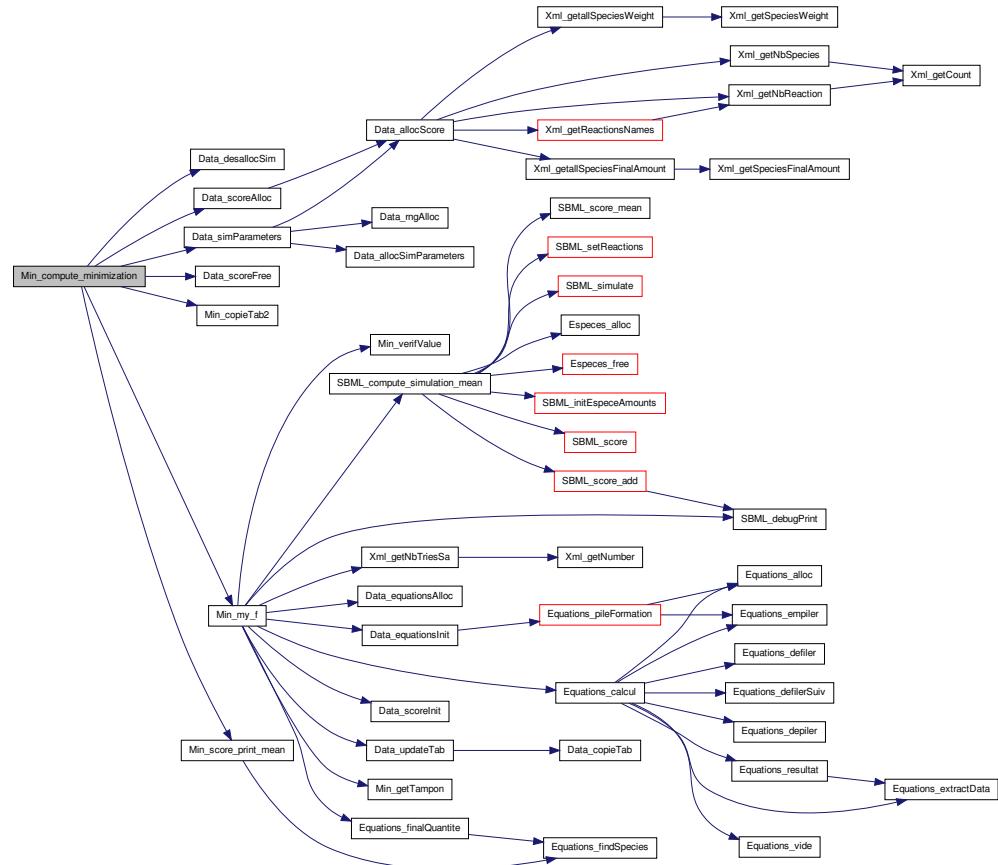
<i>allone</i>	struct <a href="#">ListParameters</a>
<i>fluxRatio</i>	Ratio parameters
<i>result_tab</i>	Result table
<i>files_path</i>	List of paths
<i>number_arg</i>	Number of simulation
<i>debug</i>	Debug flag

Definition at line 274 of file `gsl_min.c`.

References Data\_desallocSim(), Data\_scoreAlloc(), Data\_scoreFree(), Data\_simParameters(), ListParameters::interest\_parameters, Min\_copieTab2(), Min\_my\_f(), Min\_score\_print\_mean(), ListParameters::nb\_couples, ListParameters::nb\_parameters, ListParameters::parameters, and SimParameters::y.

Referenced by `Mpi_slave()`.

Here is the call graph for this function:



#### 4.4.2.2 int Min\_copieTab2 ( double \* *x*, int *current*, int *debut*, int *fin* )

Copy table of reaction parameters.

##### Author

Amine Ghozlane

##### Parameters

<i>x</i>	Short table of reaction parameters
<i>current</i>	Line
<i>debut</i>	Beginning
<i>fin</i>	End

##### Returns

Number of copied element

Definition at line 68 of file gsl\_min.c.

References SimParameters::y.

Referenced by Min\_compute\_minimization().

#### 4.4.2.3 int Min\_copieTab3 ( gsl\_multimin\_fminimizer \* *s*, int *current*, int *debut*, int *fin* )

Copy short table of parameter into the larger table.

##### Author

Amine Ghozlane

##### Parameters

<i>s</i>	Minimizer parameter
<i>current</i>	Line
<i>debut</i>	Beginning
<i>fin</i>	End

##### Returns

Number of copied element

Definition at line 90 of file gsl\_min.c.

References SimParameters::y.

#### 4.4.2.4 void Min\_getTampon ( double \* *energie\_temp*, double *energie* )

Enter in the program for standard deviation.

**Author**

Amine Ghozlane

**Parameters**

<i>energie_- temp</i>	Current energy
<i>energie</i>	Best energy

Definition at line 208 of file `gsl_min.c`.

References `ListParameters::nb_parameters`, `SimParameters::out`, `Score::quantite`, `Score::taille`, and `SimParameters::y`.

Referenced by `Min_my_f()`.

#### 4.4.2.5 double Min\_my\_f ( const `gsl_vector` \* *v*, void \* *params* )

Enter in the program for standard deviation.

**Author**

Amine Ghozlane

**Parameters**

<i>v</i>	Vector of reaction parameters
<i>params</i>	Unused parameter define by GSL

**Returns**

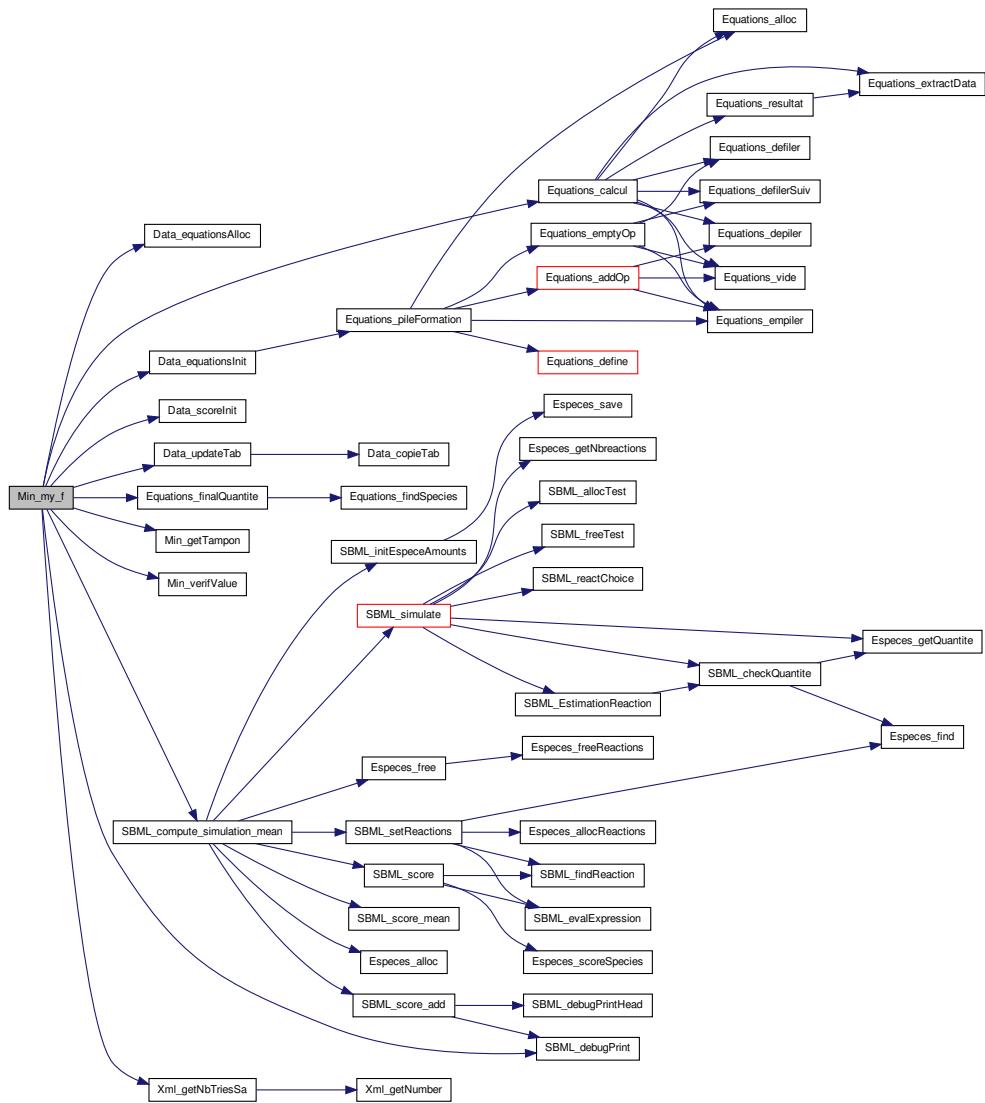
Energy value

Definition at line 149 of file `gsl_min.c`.

References `ListParameters::banned`, `ListParameters::conf`, `Data_equationsAlloc()`, `Data_-equationsInit()`, `Data_scoreInit()`, `Data_updateTab()`, `SimParameters::debugFile`, `Equations_-calcul()`, `Equations_finalQuantite()`, `Min_getTampon()`, `Min_verifValue()`, `ListParameters::model`, `Score::name`, `ListParameters::nb_banned`, `ListParameters::nb_couples`, `ListParameters::nb_-equations`, `ListParameters::nb_parameters`, `Score::nb_species`, `SimParameters::out`, `ListParameters::parameters`, `SimParameters::pile`, `Score::quantite`, `SimParameters::r`, `SBML_-compute_simulation_mean()`, `SBML_debugPrint()`, `Score::species`, `Score::species_amount`, `Score::species_weight`, `Score::taille`, `Score::tailleSpecies`, `Xml_getNbTriesSa()`, and `SimParameters::y`.

Referenced by `Min_compute_minimization()`.

Here is the call graph for this function:



#### 4.4.2.6 void Min\_score\_print\_mean ( double \* result\_tab )

Save the minimization result.

## Author

Amine Ghozlane

**Parameters**

<i>result_tab</i>	Result table
-------------------	--------------

Definition at line 230 of file `gsl_min.c`.

References `Equations_findSpecies()`, `Score::name`, `ListParameters::nb_parameters`, `Score::nb_species`, `SimParameters::out`, `Score::quantite`, `Score::species`, `Score::species_amount`, `Score::taille`, `Score::tailleSpecies`, and `SimParameters::y`.

Referenced by `Min_compute_minimization()`.

Here is the call graph for this function:



#### 4.4.2.7 void Min\_verifValue ( `gsl_vector * v`, `double * x`, `int debut`, `int max` )

Checking reaction ratio parameters.

**Author**

Amine Ghozlane

**Parameters**

<i>v</i>	Vector of reaction parameters
<i>x</i>	Short table of reaction parameters
<i>debut</i>	Beginning
<i>max</i>	End

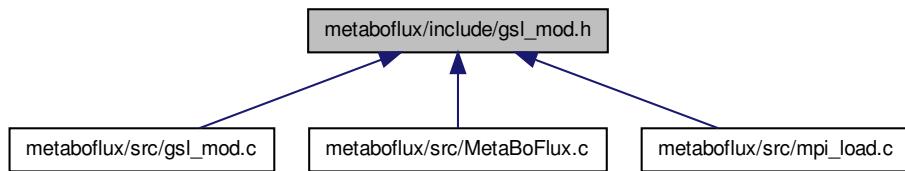
Definition at line 114 of file `gsl_min.c`.

Referenced by `Min_my_f()`.

## 4.5 metaboflux/include/gsl\_mod.h File Reference

Compute the modeling of scenarii.

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



## Functions

- void [Mod\\_score\\_print\\_mean](#) (pListParameters, pSimParameters, double \*, double, int, int)  
*Print the mean score.*
- void [Mod\\_compute\\_modeling](#) (pListParameters, double \*, double \*, char \*\*, int, int)  
*Compute the modeling.*

### 4.5.1 Detailed Description

Compute the modeling of scenarii. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflu>)  
 Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

{Amine Ghozlane}

#### Version

2.0

#### Date

9 novembre 2009

Definition in file [gsl\\_mod.h](#).

### 4.5.2 Function Documentation

4.5.2.1 void Mod\_compute\_modeling ( *pListParameters a*, double \* *fluxRatio*, double \* *result\_tab*, char \*\* *files\_path*, int *number\_group*, int *group*, int *debug* )

Compute the modeling.

#### Author

Amine Ghozlane

#### Parameters

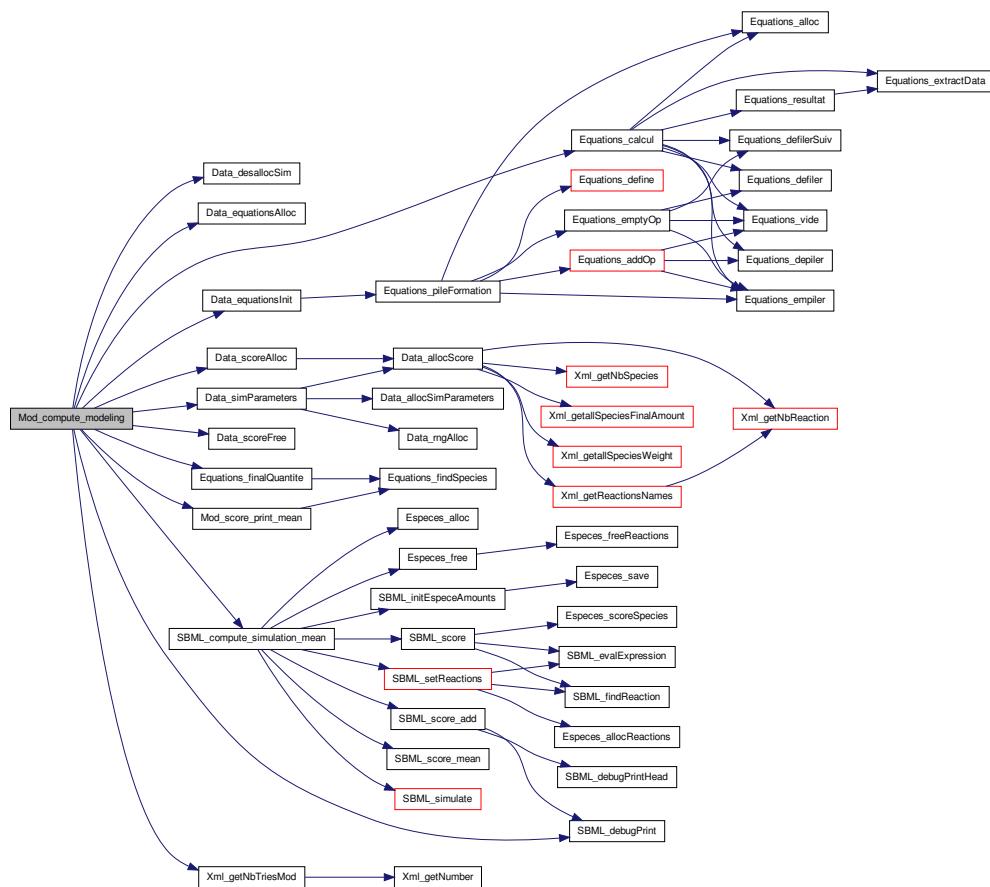
<i>a</i>	struct <a href="#">ListParameters</a>
<i>fluxRatio</i>	Ratio parameters
<i>result_tab</i>	Result table
<i>files_path</i>	List of paths
<i>number_group</i>	Number of the group
<i>group</i>	Group flag
<i>debug</i>	Debug flag

Definition at line 109 of file `gsl_mod.c`.

References `ListParameters::banned`, `ListParameters::conf`, `Data_desallocSim()`, `Data_equationsAlloc()`, `Data_equationsInit()`, `Data_scoreAlloc()`, `Data_scoreFree()`, `Data_simParameters()`, `SimParameters::debugFile`, `Equations_calcul()`, `Equations_finalQuantite()`, `Mod_score_print_mean()`, `ListParameters::model`, `Score::name`, `ListParameters::nb_banned`, `ListParameters::nb_equations`, `ListParameters::nb_parameters`, `Score::nb_species`, `SimParameters::out`, `SimParameters::pile`, `Score::quantite`, `SimParameters::r`, `SBML_compute_simulation_mean()`, `SBML_debugPrint()`, `Score::species`, `Score::species_amount`, `Score::species_weight`, `Score::taille`, `Score::tailleSpecies`, `Xml_getNbTriesMod()`, and `SimParameters::y`.

Referenced by `Mpi_slave()`.

Here is the call graph for this function:



**4.5.2.2 void Mod\_score\_print\_mean ( pListParameters a, pSimParameters simu, double \* result\_tab, double energie, int number\_group, int group )**

Print the mean score.

## Author

Amine Ghozlane

## Parameters

<i>a</i>	Global parameters
<i>simu</i>	Simulation parameters
<i>result_tab</i>	Result table
<i>energie</i>	Energy of the simulation

<code>number_- group</code>	Number of simulation
<code>group</code>	Number of the group (if a group is simulated)

Definition at line 58 of file `gsl_mod.c`.

References `Equations_findSpecies()`, `Score::name`, `ListParameters::nb_parameters`, `Score::nb_species`, `SimParameters::out`, `Score::quantite`, `Score::species`, `Score::species_amount`, `Score::taille`, `Score::tailleSpecies`, and `SimParameters::y`.

Referenced by `Mod_compute_modeling()`.

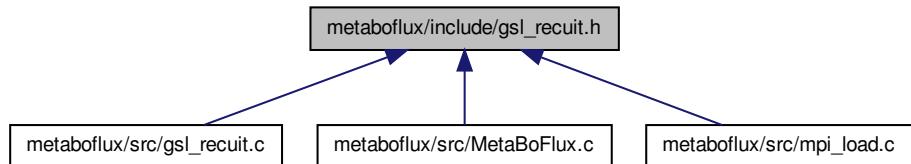
Here is the call graph for this function:



## 4.6 metaboflux/include/gsl\_recuit.h File Reference

Compute the simulated annealing.

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



### Functions

- double [Recuit\\_energyFunction](#) (void \*)  
*Simulate the petri net and compute the energy.*
- double [Recuit\\_metricDistance](#) (void \*, void \*)

*Compute the distance between the two table.*

- void [Recuit\\_takeStep](#) (const gsl\_rng \*, void \*, double)

*Take a step through the solution space TSP.*

- void [Recuit\\_printPosition](#) (void \*)

*Print the table of reaction parameters.*

- FILE \* [Recuit\\_redirectionFlux](#) (xmlConfig\_t \*, FILE \*, char \*\*, int)

*Redirect stdout flux.*

- void [Recuit\\_verifParameters](#) (double \*, const gsl\_rng \*, int, int)

*Choice of parameters and verification of their value.*

- void [Recuit\\_verifParameters\\_2](#) (double \*, double \*, const gsl\_rng \*, int, int, double)

*Choice of parameters and verification of their value.*

- void [Recuit\\_defParametre](#) (double \*, const gsl\_rng \*)

*Definition of the initial parameters of the simulation.*

- void [Recuit\\_printParametre](#) (double \*)

*Print initial parameters.*

- void [Recuit\\_compute\\_recuit](#) (char \*\*, int, int, pListParameters, gsl\_siman\_params\_t)

*Compute the simulated annealing.*

#### 4.6.1 Detailed Description

Compute the simulated annealing. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflu>)  
Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

##### Author

{Amine Ghozlane}

##### Version

2.0

##### Date

9 novembre 2009

Definition in file [gsl\\_recuit.h](#).

## 4.6.2 Function Documentation

4.6.2.1 void Recuit\_compute\_recuit ( *char \*\* files\_path, int debug, int number, pListParameters allone, gsl\_siman\_params\_t params* )

Compute the simulated annealing.

### Author

Amine Ghozlane

### Parameters

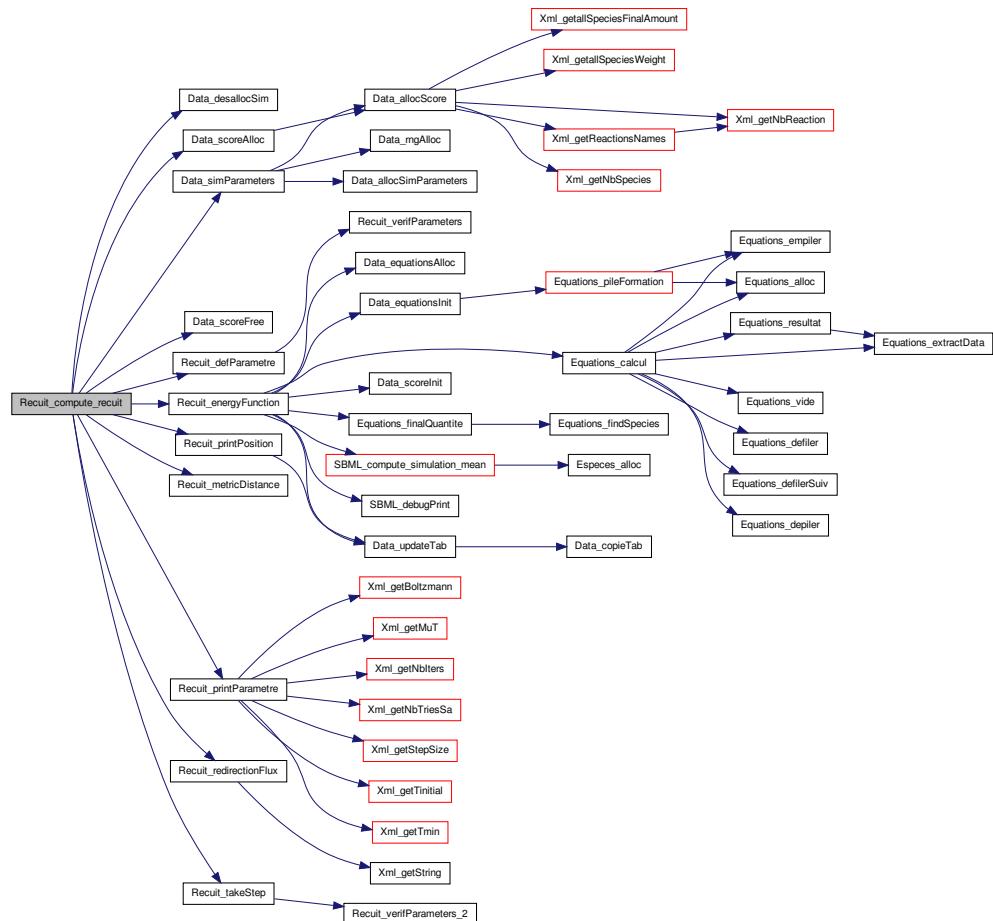
<i>files_path</i>	List of paths
<i>debug</i>	Debug flag
<i>number</i>	Number of the simulation
<i>allone</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>params</i>	Gsl parameters

Definition at line 304 of file `gsl_recuit.c`.

References `ListParameters::conf`, `Data_desallocSim()`, `Data_scoreAlloc()`, `Data_scoreFree()`, `Data_simParameters()`, `ListParameters::interest_parameters`, `SimParameters::r`, `Recuit_defParametre()`, `Recuit_energyFunction()`, `Recuit_metricDistance()`, `Recuit_printParametre()`, `Recuit_printPosition()`, `Recuit_redirectionFlux()`, and `Recuit_takeStep()`.

Referenced by `Mpi_slave()`.

Here is the call graph for this function:



#### 4.6.2.2 void Recuit\_defParametre ( double \* *x\_initial*, const gsl\_rng \* *r* )

Definition of the initial parameters of the simulation.

##### Author

Amine Ghozlane

##### Parameters

<i>x_initial</i>	table of reaction parameters
<i>r</i>	Random number generator

Definition at line 261 of file `gsl_recuit.c`.

References `ListParameters::nb_couples`, `ListParameters::parameters`, and `Recuit_verifParameters()`.

Referenced by `Recuit_compute_recuit()`.

Here is the call graph for this function:



#### 4.6.2.3 double Recuit\_energyFunction ( void \* xp )

Simulate the petri net and compute the energy.

`double Recuit_energyFunction(void *xp)`

##### Author

Amine Ghozlane

##### Parameters

<code>xp</code>	Short table of reaction parameters
-----------------	------------------------------------

##### Returns

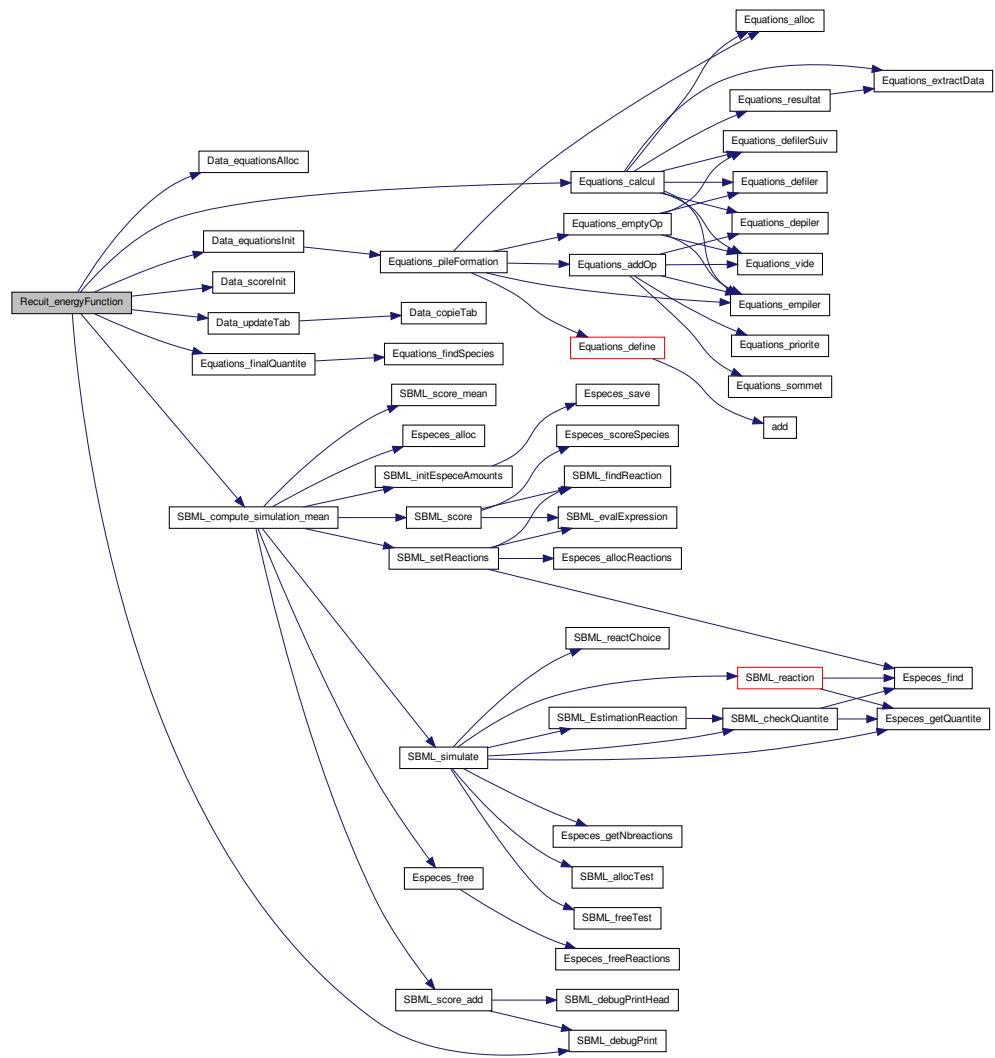
Energy value

Definition at line 65 of file `gsl_recuit.c`.

References `ListParameters::banned`, `Data_equationsAlloc()`, `Data_equationsInit()`, `Data_scoreInit()`, `Data_updateTab()`, `SimParameters::debugFile`, `Equations_calcul()`, `Equations_finalQuantite()`, `ListParameters::model`, `Score::name`, `ListParameters::nb_banned`, `ListParameters::nb_equations`, `Score::nb_species`, `ListParameters::nb_triesSa`, `SimParameters::out`, `SimParameters::pile`, `Score::quantite`, `SimParameters::r`, `SBML_compute_simulation_mean()`, `SBML_debugPrint()`, `Score::species`, `Score::species_amount`, `Score::species_weight`, `Score::taille`, `Score::tailleSpecies`, and `SimParameters::y`.

Referenced by `Recuit_compute_recuit()`.

Here is the call graph for this function:



4.6.2.4 double Recuit\_metricDistance ( void \* *xp*, void \* *yp* )

Compute the distance between the two table.

## Author

Amine Ghozlane

**Parameters**

<i>xp</i>	Past short table of reaction parameters
<i>yp</i>	New short table of reaction parameters

**Returns**

Distance between the two table

Definition at line 115 of file `gsl_recuit.c`.

References `ListParameters::interest_parameters`.

Referenced by `Recuit_compute_recuit()`.

**4.6.2.5 void Recuit\_printParametre ( double \* *x\_initial* )**

Print initial parameters.

**Author**

Amine Ghozlane

**Parameters**

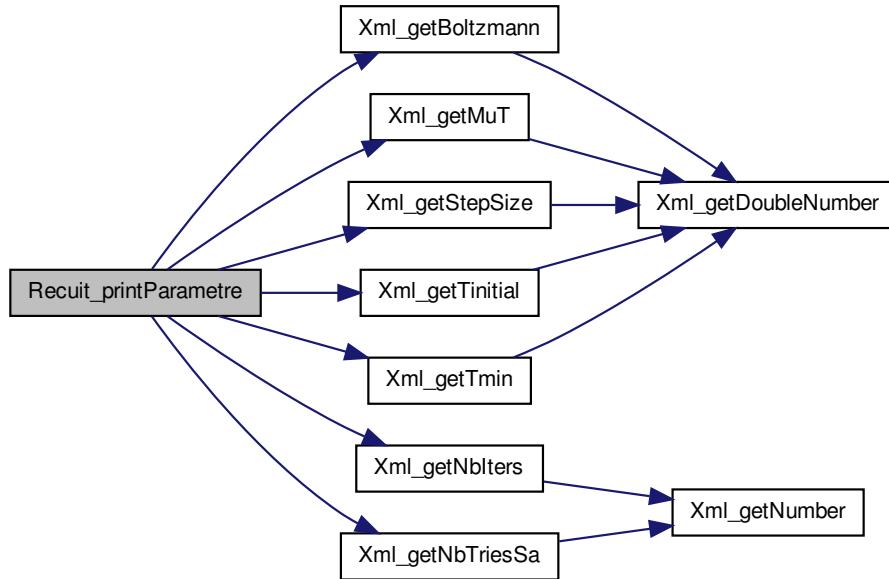
<i>x_initial</i>	table of reaction parameters
------------------	------------------------------

Definition at line 279 of file `gsl_recuit.c`.

References `ListParameters::conf`, `ListParameters::interest_parameters`, `Xml_getBoltzmann()`, `Xml_getMuT()`, `Xml_getNbIter()`, `Xml_getNbTriesSa()`, `Xml_getStepSize()`, `Xml_getTinitial()`, and `Xml_getTmin()`.

Referenced by `Recuit_compute_recuit()`.

Here is the call graph for this function:



#### 4.6.2.6 void Recuit.printPosition ( void \* xp )

Print the table of reaction parameters.

##### Author

Amine Ghozlane

##### Parameters

<code>xp</code>	Short table of reaction parameters
-----------------	------------------------------------

Definition at line 158 of file `gsl_recuit.c`.

References `Data_updateTab()`, `ListParameters::nb_parameters`, and `SimParameters::y`.

Referenced by `Recuit_compute_recuit()`.

Here is the call graph for this function:



**4.6.2.7 FILE\* Recuit\_redirectionFlux ( xmlConfig\_t \* *conf*, FILE \* *f*, char \*\* *files\_path*, int *number* )**

Redirect stdout flux.

#### Author

Amine Ghozlane

#### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>f</i>	File name
<i>files_path</i>	List of paths
<i>number</i>	Index of out repertory

#### Returns

Out flux

Definition at line 184 of file [gsl\\_recuit.c](#).

References [Xml\\_getString\(\)](#).

Referenced by [Recuit\\_compute\\_recuit\(\)](#).

Here is the call graph for this function:



#### 4.6.2.8 void Recuit\_takeStep ( const gsl\_rng \* *r*, void \* *xp*, double *step\_size* )

Take a step through the solution space TSP.

##### Author

Amine Ghozlane

##### Parameters

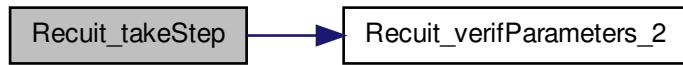
<i>r</i>	Random number generator
<i>xp</i>	Short table of reaction parameters
<i>step_size</i>	Size of step

Definition at line 136 of file gsl\_recuit.c.

References ListParameters::interest\_parameters, ListParameters::nb\_couples, ListParameters::parameters, and Recuit\_verifParameters\_2().

Referenced by Recuit\_compute\_recuit().

Here is the call graph for this function:



#### 4.6.2.9 void Recuit\_verifParameters ( double \* *x\_initial*, const gsl\_rng \* *r*, int *debut*, int *max* )

Choice of parameters and verification of their value.

##### Author

Amine Ghozlane

##### Parameters

<i>x_initial</i>	table of reaction parameters
<i>r</i>	Random number generator
<i>debut</i>	Beginning
<i>max</i>	End

Definition at line 207 of file gsl\_recuit.c.

Referenced by Recuit\_defParametre().

---

```
4.6.2.10 void Recuit_verifParameters_2( double * new_x, double * old_x, const gsl_rng * r, int debut, int max, double step_size )
```

Choice of parameters and verification of their value.

#### Author

Amine Ghozlane

#### Parameters

<i>new_x</i>	table of reaction parameters
<i>old_x</i>	table of reaction parameters
<i>r</i>	Random number generator
<i>debut</i>	Beginning
<i>max</i>	End
<i>step_size</i>	Size of the step

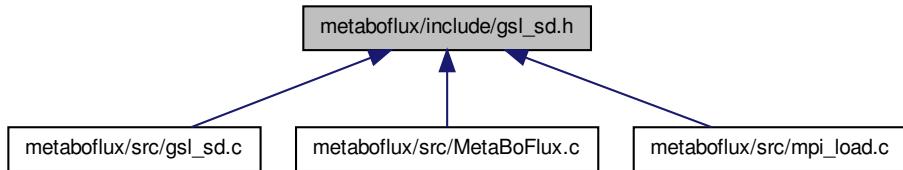
Definition at line 235 of file `gsl_recuit.c`.

Referenced by `Recuit_takeStep()`.

## 4.7 metaboflux/include/gsl\_sd.h File Reference

Compute the standard deviation analysis of the simulations.

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



#### Functions

- double [Sd\\_compute\\_simulation \(pListParameters, pSimParameters\)](#)  
*Enter in the program for standard deviation.*
- void [Sd\\_compute\\_standard\\_deviation \(pListParameters, double \\*, double \\*, char \\*\\*, int, int\)](#)  
*Compute the standard deviation.*

### 4.7.1 Detailed Description

Compute the standard deviation analysis of the simulations. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflux/>) Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

{Amine Ghozlane}

#### Version

2.0

#### Date

9 novembre 2009

Definition in file [gsl\\_sd.h](#).

### 4.7.2 Function Documentation

#### 4.7.2.1 double Sd\_compute\_simulation ( **pListParameters** *all*, **pSimParameters** *sim* )

Enter in the program for standard deviation.

#### Author

Amine Ghozlane

#### Parameters

<i>all</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>sim</i>	Simulation parameters : struct <a href="#">SimParameters</a>

#### Returns

Energy value

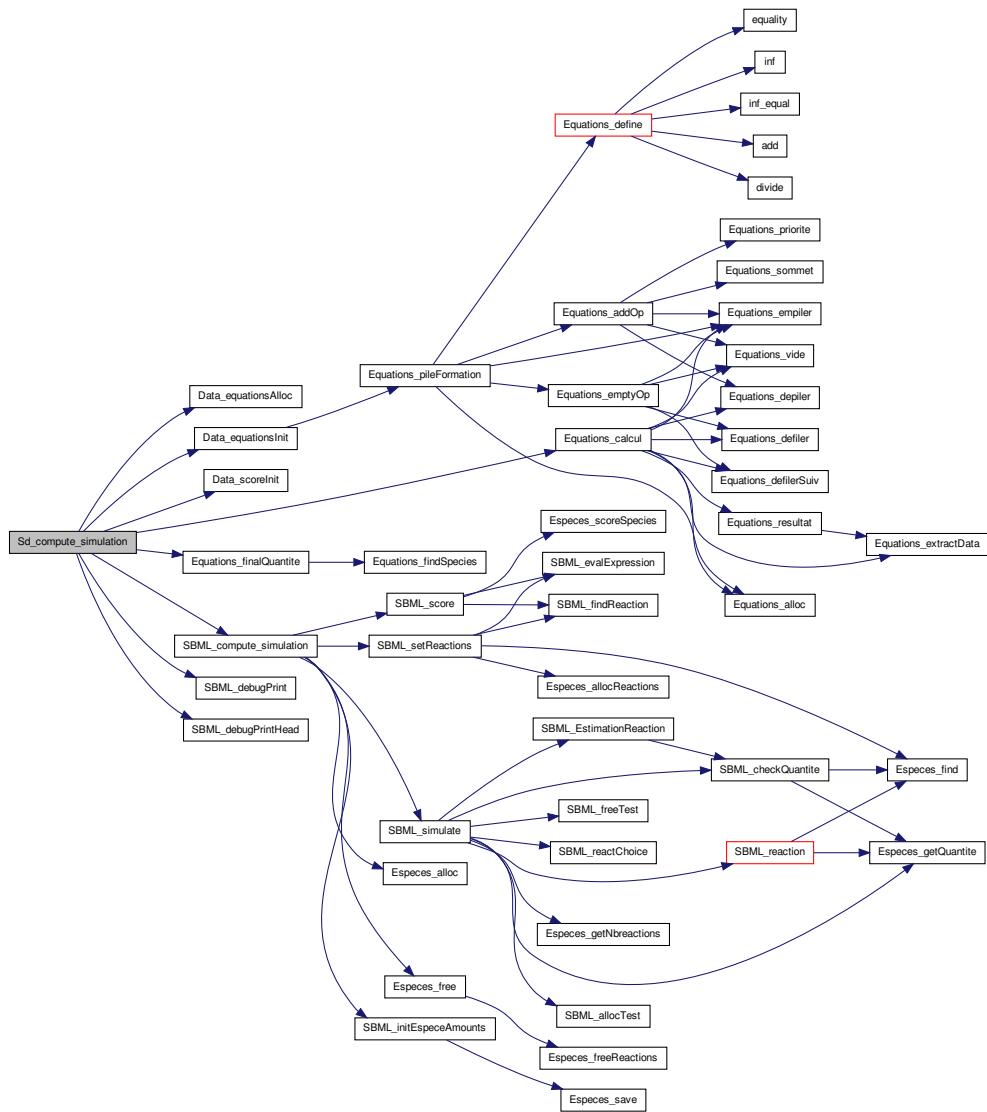
Definition at line 56 of file [gsl\\_sd.c](#).

References [ListParameters::banned](#), [Data\\_equationsAlloc\(\)](#), [Data\\_equationsInit\(\)](#), [Data\\_scoresInit\(\)](#), [SimParameters::debugFile](#), [Equations\\_calcul\(\)](#), [Equations\\_finalQuantite\(\)](#),

ListParameters::model, Score::name, ListParameters::nb\_banned, ListParameters::nb\_-equations, Score::nb\_species, SimParameters::out, SimParameters::pile, Score::quantite, SimParameters::r, SBML\_compute\_simulation(), SBML\_debugPrint(), SBML\_debugPrintHead(), Score::species, Score::species\_amount, Score::species\_weight, Score::taille, Score::tailleSpecies, and SimParameters::y.

Referenced by Sd\_compute\_standard\_deviation().

Here is the call graph for this function:



4.7.2.2 void Sd\_compute\_standard\_deviation ( pListParameters a, double \* fluxRatio, double \* result\_tab, char \*\* files\_path, int number\_arg, int debug )

Compute the standard deviation.

## Author

Amine Ghozlane

## Parameters

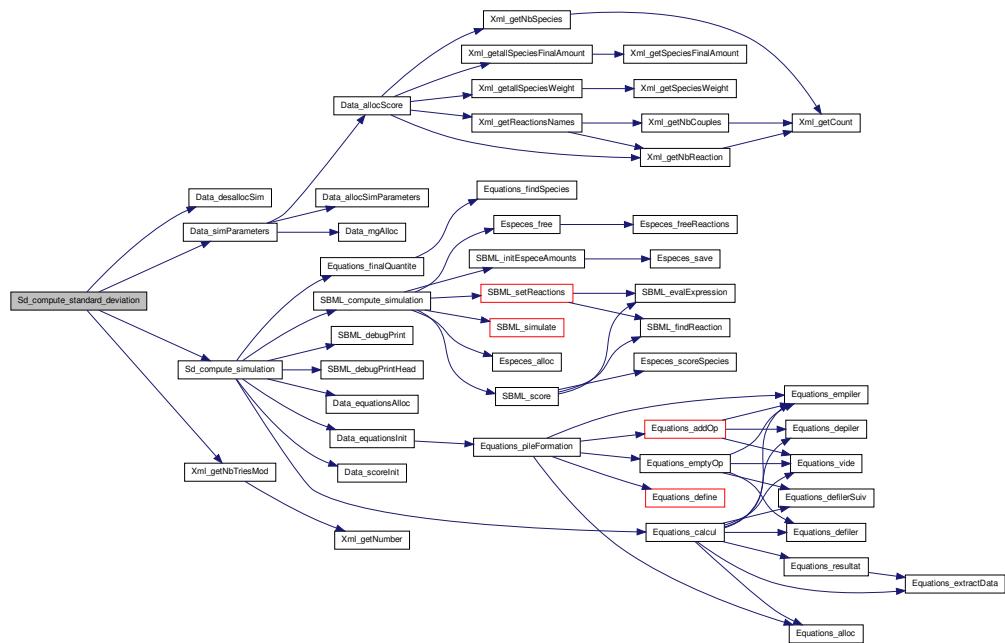
<i>a</i>	struct <a href="#">ListParameters</a>
<i>fluxRatio</i>	Ratio parameters
<i>result_tab</i>	Result table
<i>files_path</i>	List of paths
<i>number_arg</i>	Number of simulation
<i>debug</i>	Debug flag

Definition at line 106 of file `gsl_sd.c`.

References ListParameters::conf, Data\_desallocSim(), Data\_simParameters(), ListParameters::nb\_parameters, Sd\_compute\_simulation(), Xml\_getNbTriesMod(), and SimParameters::y.

Referenced by Mpi\_slave().

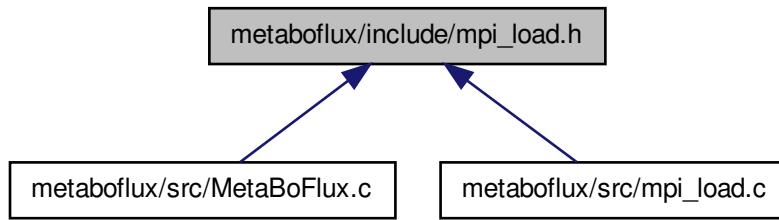
Here is the call graph for this function:



## 4.8 metaboflux/include/mpi\_load.h File Reference

Parallelize the program.

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



### Functions

- int [Mpi\\_connectInterface](#) (int)  
*Enter in the program for standard deviation.*
- void [Mpi\\_disconnectInterface](#) (int)  
*Disconnection to the interface.*
- void [Mpi\\_init](#) (int, char \*\*, int \*)  
*Enter in the program for standard deviation.*
- void [Mpi\\_master](#) (pListParameters, char \*\*, int, int, int, int, int)  
*Master process.*
- void [Mpi\\_writer](#) (pListParameters, char \*\*, int, int, int)  
*Master process.*
- int [Mpi\\_sizeResultTab](#) (pListParameters, int, int, int)  
*Determine the size of the result tab.*
- double \* [Mpi\\_allocResultTab](#) (int)  
*Allocate the result tab.*
- void [Mpi\\_writeSimFile](#) (pListParameters, FILE \*, FILE \*, double \*, int, int, int)  
*Determine the size of the result tab.*
- void [Mpi\\_writeSdFile](#) (FILE \*, double \*, int)  
*Determine the size of the result tab.*
- void [Mpi\\_slave](#) (pListParameters, char \*\*, int, int, int)
- void [Mpi\\_finalize](#) (void)  
*End MPI execution environment.*
- void [compute\\_mpi](#) (int, char \*\*, char \*\*, int, int, int, int)  
*Compute the simulated annealing through mpi.*

### 4.8.1 Detailed Description

Parallelize the program. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflux/>)  
 Copyright (C) 2010 Amine Ghazlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

{Amine Ghazlane}

#### Version

2.0

#### Date

9 novembre 2009

Definition in file [mpi\\_load.h](#).

### 4.8.2 Function Documentation

#### 4.8.2.1 void compute\_mpi ( int argc, char \*\* argv, char \*\* files\_path, int activity, int group, int debug, int port )

Compute the simulated annealing through mpi.

#### Author

Amine Ghazlane

#### Parameters

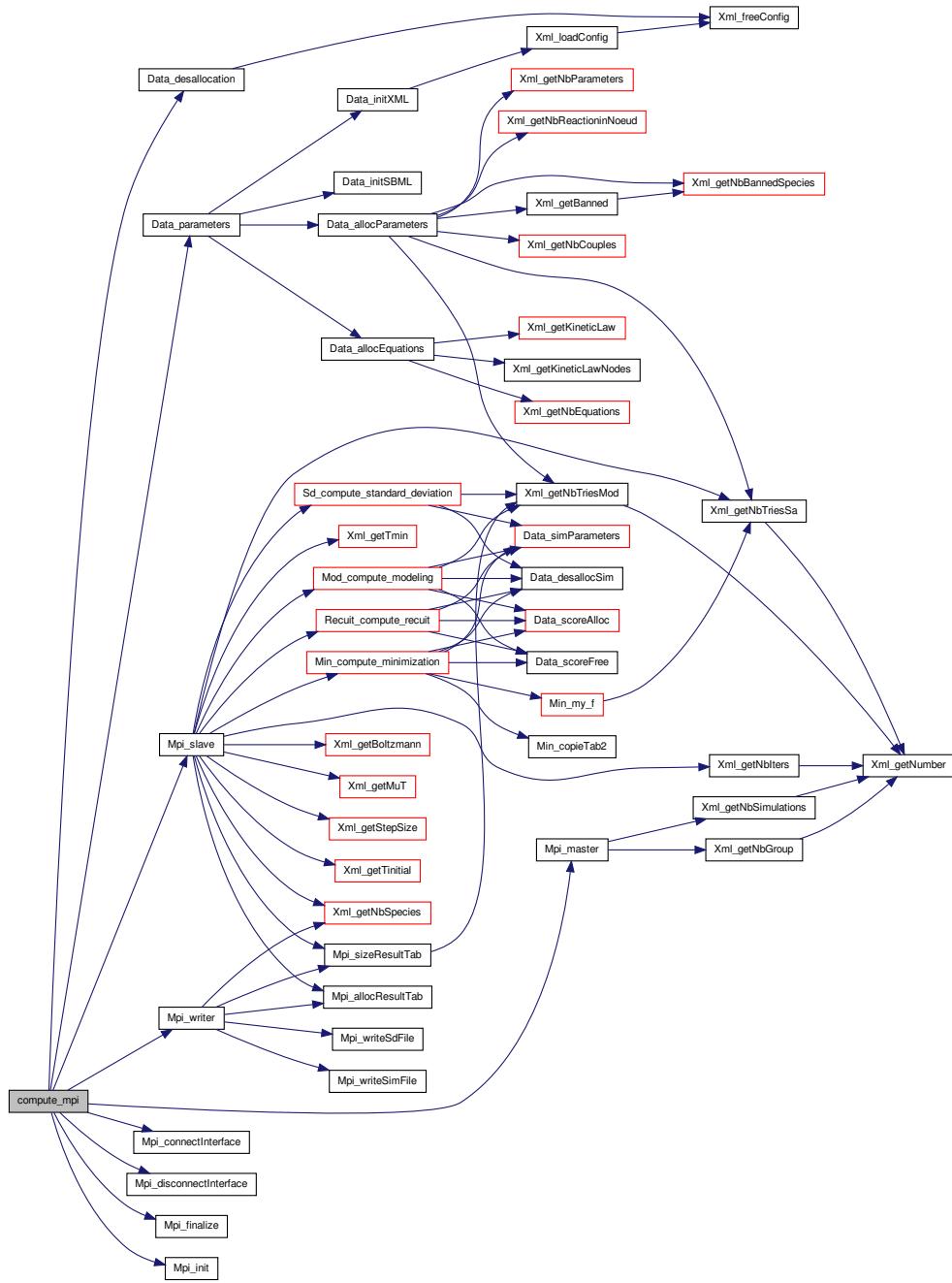
<i>argc</i>	Number of arguments
<i>argv</i>	List of arguments
<i>files_path</i>	List of paths
<i>activity</i>	Chosen activity
<i>group</i>	Group flag
<i>debug</i>	Debug flag
<i>port</i>	Interface port

Definition at line 546 of file [mpi\\_load.c](#).

References `Data_desallocation()`, `Data_parameters()`, `Mpi_connectInterface()`, `Mpi_disconnectInterface()`, `Mpi_finalize()`, `Mpi_init()`, `Mpi_master()`, `Mpi_slave()`, and `Mpi_writer()`.

Referenced by `main()`.

Here is the call graph for this function:



**4.8.2.2 double\* Mpi\_allocResultTab ( int *tailleTab* )**

Allocate the result tab.

**Author**

Amine Ghozlane

**Parameters**

<i>tailleTab</i>	Size of Result tab
------------------	--------------------

**Returns**

Address of the allocated space

Definition at line 355 of file mpi\_load.c.

Referenced by Mpi\_slave(), and Mpi\_writer().

**4.8.2.3 int Mpi\_connectInterface ( int *port* )**

Enter in the program for standard deviation.

**Author**

Amine Ghozlane

**Parameters**

<i>port</i>	Connection port
-------------	-----------------

**Returns**

Socket

Definition at line 69 of file mpi\_load.c.

Referenced by compute\_mpi().

**4.8.2.4 void Mpi\_disconnectInterface ( int *desc* )**

Disconnection to the interface.

**Author**

Amine Ghozlane

**Parameters**

<i>desc</i>	Socket
-------------	--------

Definition at line 109 of file mpi\_load.c.

Referenced by compute\_mpi().

#### 4.8.2.5 void Mpi\_finalize ( void )

End MPI execution environment.

##### Author

Amine Ghozlane

Definition at line 527 of file mpi\_load.c.

Referenced by compute\_mpi().

#### 4.8.2.6 void Mpi\_init ( int argc, char \*\* argv, int \* tab )

Enter in the program for standard deviation.

##### Author

Amine Ghozlane

##### Parameters

<i>argc</i>	Number of arguments
<i>argv</i>	List of arguments
<i>tab</i>	Table

Definition at line 123 of file mpi\_load.c.

Referenced by compute\_mpi().

#### 4.8.2.7 void Mpi\_master ( pListParameters *allone*, char \*\* *files\_path*, int *activity*, int *group*, int *myid*, int *numprocs*, int *desc* )

Master process.

##### Author

Amine Ghozlane

##### Parameters

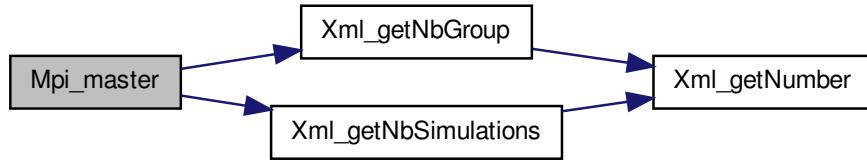
<i>allone</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>files_path</i>	List of paths
<i>activity</i>	Chosen activity
<i>group</i>	Group flag
<i>myid</i>	Id of the thread
<i>numprocs</i>	Number of thread
<i>desc</i>	Socket

Definition at line 154 of file mpi\_load.c.

References ListParameters::conf, ListParameters::nb\_parameters, Xml\_getNbGroup(), and Xml\_getNbSimulations().

Referenced by compute\_mpi().

Here is the call graph for this function:



**4.8.2.8 int Mpi\_sizeResultTab ( pListParameters *allone*, int *activity*, int *group*, int *nb\_species* )**

Determine the size of the result tab.

#### Author

Amine Ghozlane

#### Parameters

<i>allone</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>activity</i>	Chosen activity
<i>group</i>	Group flag
<i>nb_species</i>	Number of interest species

#### Returns

Size of Result tab

Definition at line 334 of file mpi\_load.c.

References ListParameters::conf, ListParameters::model, ListParameters::nb\_parameters, and Xml\_getNbTriesMod().

Referenced by Mpi\_slave(), and Mpi\_writer().

Here is the call graph for this function:



**4.8.2.9 void Mpi\_slave ( pListParameters *allone*, char \*\* *files\_path*, int *activity*, int *group*, int *debug* )**

#### Author

Amine Ghozlane

#### Parameters

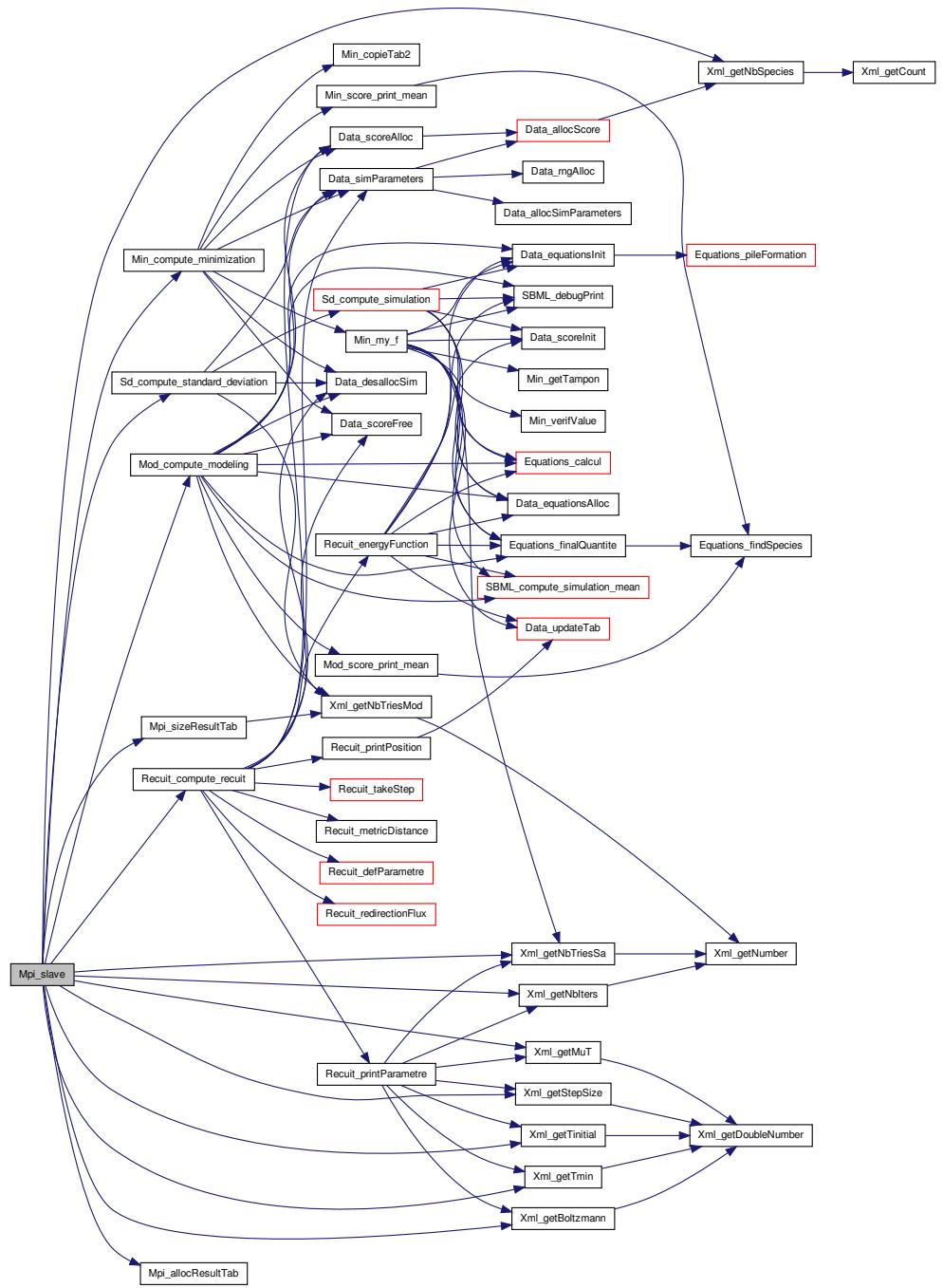
<i>allone</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>files_path</i>	List of paths
<i>activity</i>	Activity chosen
<i>group</i>	Group flag
<i>debug</i>	Debug flag

Definition at line 440 of file mpi\_load.c.

References [ListParameters::conf](#), [Min\\_compute\\_minimization\(\)](#), [Mod\\_compute\\_modeling\(\)](#), [Mpi\\_allocResultTab\(\)](#), [Mpi\\_sizeResultTab\(\)](#), [ListParameters::nb\\_parameters](#), [Recuit\\_compute\\_recuit\(\)](#), [Sd\\_compute\\_standard\\_deviation\(\)](#), [Xml\\_getBoltzmann\(\)](#), [Xml\\_getMuT\(\)](#), [Xml\\_getNbIter\(\)](#), [Xml\\_getNbSpecies\(\)](#), [Xml\\_getNbTriesSa\(\)](#), [Xml\\_getStepSize\(\)](#), [Xml\\_getTinitial\(\)](#), and [Xml\\_getTmin\(\)](#).

Referenced by [compute\\_mpi\(\)](#).

Here is the call graph for this function:



4.8.2.10 void Mpi\_writer ( pListParameters *allone*, char \*\* *files\_path*, int *activity*, int *group*, int *myid* )

Master process.

#### Author

Amine Ghozlane

#### Parameters

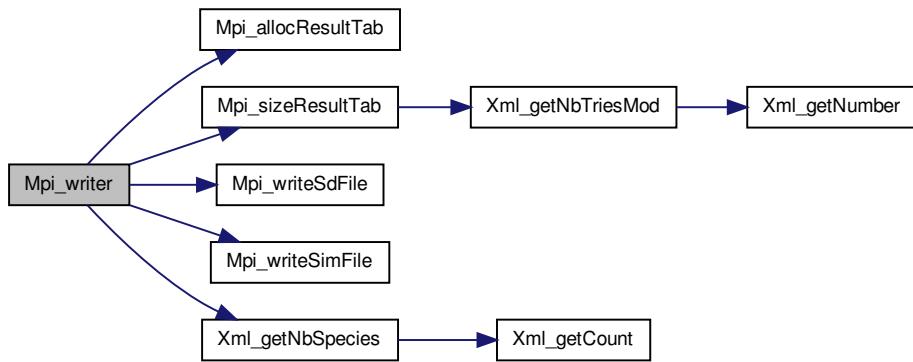
<i>allone</i>	Global parameters : struct ListParameters
<i>files_path</i>	List of paths
<i>activity</i>	Chosen activity
<i>group</i>	Group flag
<i>myid</i>	Id of the thread

Definition at line 273 of file mpi\_load.c.

References ListParameters::conf, ListParameters::model, Mpi\_allocResultTab(), Mpi\_sizeResultTab(), Mpi\_writeSdFile(), Mpi\_writeSimFile(), and Xml\_getNbSpecies().

Referenced by compute\_mpi().

Here is the call graph for this function:



4.8.2.11 void Mpi\_writeSdFile ( FILE \* *out*, double \* *result\_tab*, int *tailleTab* )

Determine the size of the result tab.

#### Author

Amine Ghozlane

**Parameters**

<i>out</i>	Result file
<i>result_tab</i>	Result table
<i>tailleTab</i>	Size of result table

Definition at line 421 of file mpi\_load.c.

Referenced by Mpi\_writer().

**4.8.2.12 void Mpi\_writeSimFile ( pListParameters *allone*, FILE \* *out*, FILE \* *logOut*, double \* *result\_tab*, int *group*, int *tailleSpecies*, int *nb\_species* )**

Determine the size of the result tab.

**Author**

Amine Ghozlane

**Parameters**

<i>allone</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>out</i>	Result file
<i>logOut</i>	Log file
<i>result_tab</i>	Result table
<i>group</i>	Group flag
<i>tailleSpecies</i>	Number of species
<i>nb_species</i>	Number of interest species

Definition at line 375 of file mpi\_load.c.

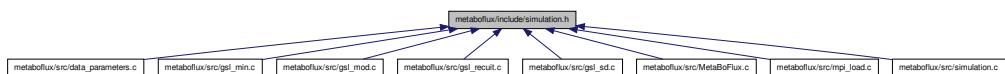
References ListParameters::nb\_parameters.

Referenced by Mpi\_writer().

## 4.9 metaboflux/include/simulation.h File Reference

Simulate a petri net.

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



## Data Structures

- struct [TestReaction](#)

*Structure used to test reactions.*

- struct **Score**

*Structure containing all information from simulations.*

## Functions

- void **SBML\_initEspeceAmounts** (Model\_t \*, pEspecies, int)
 

*Alloc memory and initialize the struct **Especies**.*
- int **SBML\_findReaction** (char \*\*, const char \*, int)
 

*Determine if the reaction is study.*
- double **SBML\_evalExpression** (const char \*)
 

*Get the reaction ratio define in the sbml.*
- void **SBML\_setReactions** (Model\_t \*, pEspecies, pScore, double \*, int, int)
 

*Alloc memory and initialize the struct **Especies**.*
- int **SBML\_checkQuantite** (Model\_t \*, Reaction\_t \*, int, pEspecies)
 

*Determine the number of reaction for one molecule.*
- Reaction\_t \* **SBML\_reactChoice** (pEspecies, const gsl\_rng \*, int)
 

*Determine randomly the reaction to achieve for several nodes reactions.*
- void **SBML\_reaction** (Model\_t \*, pEspecies, Reaction\_t \*, int)
 

*Simulation of a discrete transision.*
- void **SBML\_allocTest** (pTestReaction, int)
 

*Alloc memory and initialize the struct **pTestReaction**.*
- void **SBML\_freeTest** (pTestReaction)
 

*Free memory of the struct **TestReaction**.*
- int **SBML\_EstimationReaction** (Model\_t \*, pTestReaction, pEspecies, int, int)
 

*Alloc memory and initialize the struct **Especies**.*
- int **SBML\_simulate** (Model\_t \*, pEspecies, const gsl\_rng \*, pTestReaction, char \*\*, int, int, int)
 

*Simulate one step of petri net.*
- void **SBML\_score** (Model\_t \*, pEspecies, pScore, double \*, int, int)
 

*Alloc memory and initialize the struct **Especies**.*
- void **SBML\_debugPrintHead** (FILE \*, int, char \*\*)
 

*Print the head of the debug file.*
- void **SBML\_debugPrint** (FILE \*, int, int, double \*, double)
 

*Print the debuf file.*
- void **SBML\_compute\_simulation** (pScore, Model\_t \*, double \*, gsl\_rng \*, char \*\*, int)
 

*Simulation of metabolic network.*
- void **SBML\_score\_add** (pScore, pScore, FILE \*)
 

*Add scores.*
- void **SBML\_score\_mean** (pScore, int)
 

*Mean quantities for score.*
- void **SBML\_compute\_simulation\_mean** (FILE \*, pScore, pScore, Model\_t \*, double \*, gsl\_rng \*, char \*\*, int, int)
 

*X time simulation of metabolic network.*

#### 4.9.1 Detailed Description

Simulate a petri net. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflux>)  
Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

{Amine Ghozlane}

**Version**

2.0

**Date**

27 octobre 2009

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

#### 4.9.2 Function Documentation

##### 4.9.2.1 void SBML\_allocTest( pTestReaction *T*, int *nbReactions* )

Alloc memory and initialize the struct pTestReaction.

**Author**

Amine Ghozlane

**Parameters**

<i>T</i>	Empty struct <a href="#">TestReaction</a>
<i>nbReactions</i>	Number of reactions

Definition at line 279 of file simulation.c.

References [TestReaction::minStepTab](#), and [TestReaction::tabReactions](#).

Referenced by [SBML\\_simulate\(\)](#).

**4.9.2.2 int SBML\_checkQuantite ( Model\_t \* mod, Reaction\_t \* react, int nbEspecies, pEspecies molecules )**

Determine the number of reaction for one molecule.

#### Author

Amine Ghozlane

#### Parameters

<i>mod</i>	Model of the SBML file
<i>react</i>	Reaction id
<i>nbEspecies</i>	Number of molecules
<i>molecules</i>	Struct <a href="#">Especies</a>

#### Returns

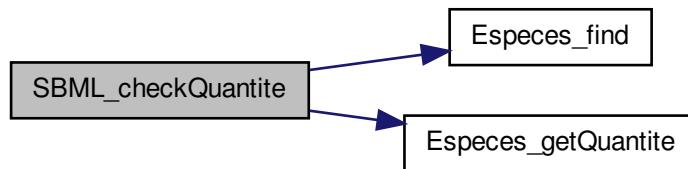
Number of reaction for one molecule

Definition at line 158 of file simulation.c.

References [Especies\\_find\(\)](#), and [Especies\\_getQuantite\(\)](#).

Referenced by [SBML\\_EstimationReaction\(\)](#), and [SBML\\_simulate\(\)](#).

Here is the call graph for this function:



**4.9.2.3 void SBML\_compute\_simulation ( pScore result, Model\_t \* mod, double \* reactions\_ratio, gsl\_rng \* r, char \*\* banned, int nbBanned )**

Simulation of metabolic network.

#### Author

Amine Ghozlane

#### Parameters

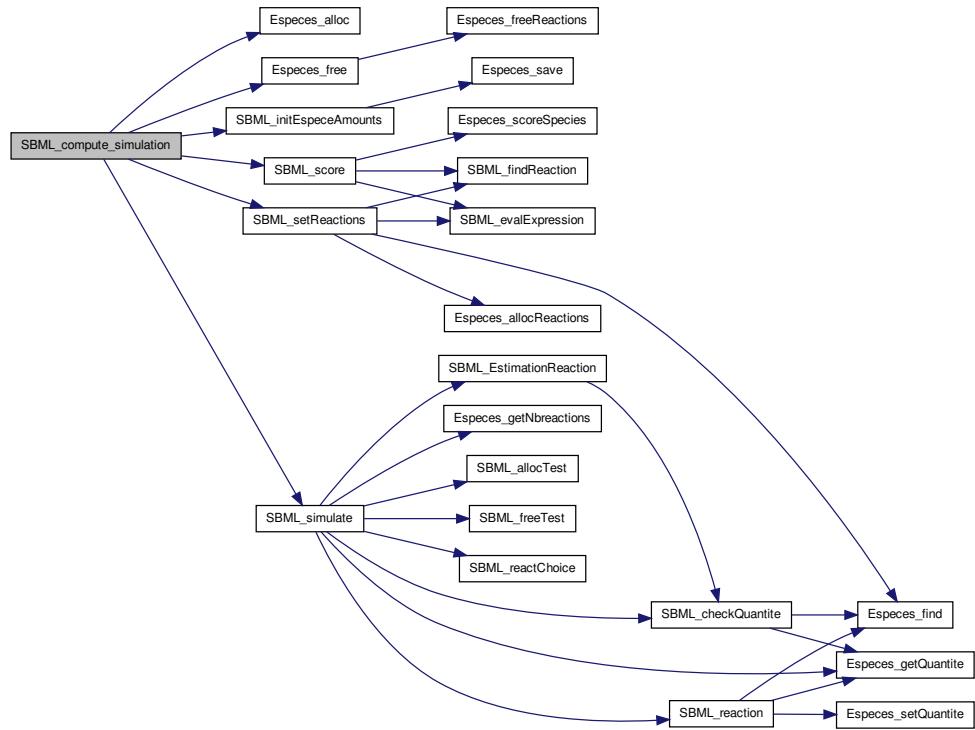
<i>result</i>	Struct Score
<i>mod</i>	Model of the SBML file
<i>reactions_-ratio</i>	List of computed reaction ratio
<i>r</i>	Random number generator
<i>banned</i>	List of banned compound
<i>nbBanned</i>	Number of banned compound

Definition at line 515 of file simulation.c.

References Espices\_alloc(), Espices\_free(), SBML\_initEspeceAmounts(), SBML\_score(), SBML\_setReactions(), SBML\_simulate(), Score::tailleReactions, and Score::tailleSpecies.

Referenced by Sd\_compute\_simulation().

Here is the call graph for this function:



```
4.9.2.4 void SBML_compute_simulation_mean ( FILE * debugFile, pScore result, pScore
result_temp, Model_t * mod, double * reactions_ratio, gsl_rng * r, char ** banned, int
nbBanned, int nb_simulation )
```

X time simulation of metabolic network.

#### Author

Amine Ghozlane

#### Parameters

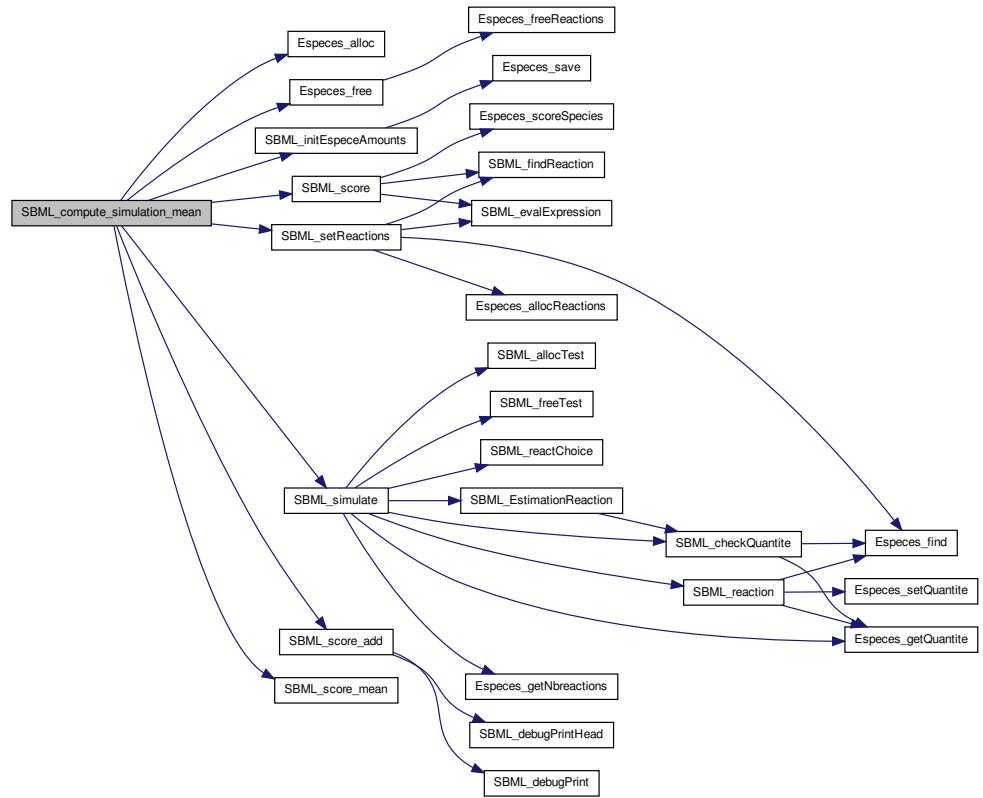
<i>debugFile</i>	File use for debug
<i>result</i>	Struct <a href="#">Score</a> used for all the simulation
<i>result_temp</i>	Struct <a href="#">Score</a> used at each simulation step
<i>mod</i>	Model of the SBML file
<i>reactions_-ratio</i>	List of computed reaction ratio
<i>r</i>	Random number generator
<i>banned</i>	List of banned compound
<i>nbBanned</i>	Number of banned compound
<i>nb_-simulation</i>	Number of simulation step

Definition at line 614 of file simulation.c.

References [Especies\\_alloc\(\)](#), [Especies\\_free\(\)](#), [SBML\\_initEspeceAmounts\(\)](#), [SBML\\_score\(\)](#), [SBML\\_score\\_add\(\)](#), [SBML\\_score\\_mean\(\)](#), [SBML\\_setReactions\(\)](#), and [SBML\\_simulate\(\)](#).

Referenced by [Min\\_my\\_f\(\)](#), [Mod\\_compute\\_modeling\(\)](#), and [Recuit\\_energyFunction\(\)](#).

Here is the call graph for this function:



**4.9.2.5 void SBML\_debugPrint ( FILE \* *debugFile*, int *tailleSpecies*, int *taille*, double \* *quantite*, double *result* )**

Print the debuf file.

#### Author

Amine Ghozlane

#### Parameters

<i>debugFile</i>	File use for debug
<i>tailleSpecies</i>	Number of molecules
<i>taille</i>	Number of molecules/reactions
<i>quantite</i>	Quantity of molecules/reactions
<i>result</i>	Energy value

Definition at line 486 of file simulation.c.

Referenced by Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), SBML\_-score\_add(), and Sd\_compute\_simulation().

#### 4.9.2.6 void SBML\_debugPrintHead ( FILE \* *debugFile*, int *taille*, char \*\* *name* )

Print the head of the debug file.

##### Author

Amine Ghozlane

##### Parameters

<i>debugFile</i>	File use for debug
<i>taille</i>	Number of molecules/reactions
<i>name</i>	List of molecules/reactions

Definition at line 465 of file simulation.c.

Referenced by SBML\_score\_add(), and Sd\_compute\_simulation().

#### 4.9.2.7 int SBML\_EstimationReaction ( Model\_t \* *mod*, pTestReaction *T*, pEspecies *molecules*, int *ref*, int *nbEspecies* )

Alloc memory and initialize the struct [Especies](#).

##### Author

Amine Ghozlane

##### Parameters

<i>mod</i>	Model of the SBML file
<i>T</i>	Struct <a href="#">TestReaction</a> gives data on reaction
<i>molecules</i>	Struct <a href="#">Especies</a>
<i>ref</i>	Number reference of one molecule
<i>nbEspecies</i>	Number of molecules

##### Returns

Estimated number of feasible step by reaction

Definition at line 322 of file simulation.c.

References Reaction::link, TestReaction::minStepTab, SBML\_checkQuantite(), Reaction::suivant, Espieces::system, and TestReaction::tabReactions.

Referenced by SBML\_simulate().

Here is the call graph for this function:



#### 4.9.2.8 double SBML\_evalExpression ( const char \* *formule* )

Get the reaction ratio define in the sbml.

##### Author

Amine Ghozlane

##### Parameters

<i>formule</i>	Formule SBML
----------------	--------------

##### Returns

Return double value of the constraint

Definition at line 88 of file simulation.c.

Referenced by SBML\_score(), and SBML\_setReactions().

#### 4.9.2.9 int SBML\_findReaction ( char \*\* *reaction*, const char \* *react*, int *nb\_reaction* )

Determine if the reaction is study.

##### Author

Amine Ghozlane

##### Parameters

<i>reaction</i>	List of reactions
<i>react</i>	Reaction of interest
<i>nb_reaction</i>	Number of reactions

##### Returns

Number of the molecules if it's study

Definition at line 69 of file simulation.c.

Referenced by SBML\_score(), and SBML\_setReactions().

#### 4.9.2.10 void SBML\_freeTest ( pTestReaction *T* )

Free memory of the struct [TestReaction](#).

##### Author

Amine Ghozlane

##### Parameters

<i>T</i>	Struct <a href="#">TestReaction</a> gives data on reaction
----------	--

Definition at line 304 of file simulation.c.

References TestReaction::minStepTab, and TestReaction::tabReactions.

Referenced by SBML\_simulate().

#### 4.9.2.11 void SBML\_initEspecieAmounts ( Model\_t \* *mod*, pEspecies *molecules*, int *nbEspecies* )

Alloc memory and initialize the struct [Especies](#).

##### Author

Amine Ghozlane

##### Parameters

<i>mod</i>	Model of the SBML file
<i>molecules</i>	Struct <a href="#">Especies</a>
<i>nbEspecies</i>	Number of molecules

Definition at line 46 of file simulation.c.

References [Especies\\_save\(\)](#).

Referenced by SBML\_compute\_simulation(), and SBML\_compute\_simulation\_mean().

Here is the call graph for this function:



#### 4.9.2.12 Reaction\_t\* SBML\_reactChoice ( pEspecies molecules, const gsl\_rng \* r, int ref )

Determine randomly the reaction to achieve for several nodes reactions.

##### Author

Amine Ghozlane

##### Parameters

<i>molecules</i>	Struct <a href="#">Especies</a>
<i>r</i>	Random number generator
<i>ref</i>	Number reference of one molecule

##### Returns

Id of the selected reaction

Definition at line 206 of file simulation.c.

References Reaction::link, Reaction::suivant, and Especies::system.

Referenced by SBML\_simulate().

#### 4.9.2.13 void SBML\_reaction ( Model\_t \* mod, pEspecies molecules, Reaction\_t \* react, int nbEspecies )

Simulation of a discrete transision.

##### Author

Amine Ghozlane

##### Parameters

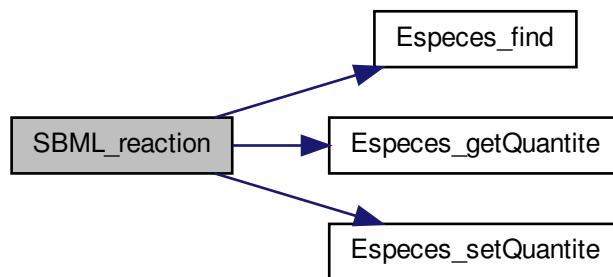
<i>mod</i>	Model of the SBML file
<i>molecules</i>	Struct <a href="#">Especies</a>
<i>react</i>	<a href="#">Reaction</a> id
<i>nbEspecies</i>	Number of molecules

Definition at line 244 of file simulation.c.

References `Especies_find()`, `Especies_getQuantite()`, and `Especies_setQuantite()`.

Referenced by `SBML_simulate()`.

Here is the call graph for this function:



**4.9.2.14 void SBML\_score ( Model\_t \* mod, pEspecies molecules, pScore result, double \* reactions\_ratio, int nbReactions, int nbEspecies )**

Alloc memory and initialize the struct [Especies](#).

#### Author

Amine Ghozlane

#### Parameters

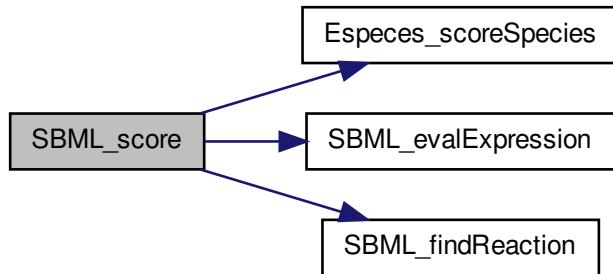
<i>mod</i>	Model of the SBML file
<i>molecules</i>	Struct <a href="#">Especies</a>
<i>result</i>	Struct <a href="#">Score</a>
<i>reactions_ratio</i>	List of computed reaction ratio
<i>nbReactions</i>	Number of reactions
<i>nbEspecies</i>	Number of molecules

Definition at line 420 of file simulation.c.

References `Especies_scoreSpecies()`, `Score::name`, `Score::nb_reaction`, `Score::quantite`, `Score::reaction`, `SBML_evalExpression()`, and `SBML_findReaction()`.

Referenced by `SBML_compute_simulation()`, and `SBML_compute_simulation_mean()`.

Here is the call graph for this function:



#### 4.9.2.15 void SBML\_score\_add ( pScore result, pScore result\_temp, FILE \* debugFile )

Add scores.

##### Author

Amine Ghozlane

##### Parameters

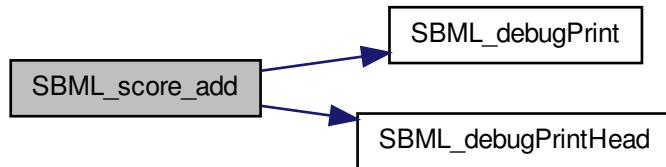
<code>result</code>	Struct <code>Score</code> used for all the simulation
<code>result_temp</code>	Struct <code>Score</code> used at each simulation step
<code>debugFile</code>	File use for debug

Definition at line 560 of file simulation.c.

References `Score::name`, `Score::quantite`, `SBML_debugPrint()`, `SBML_debugPrintHead()`, `Score::taille`, and `Score::tailleSpecies`.

Referenced by `SBML_compute_simulation_mean()`.

Here is the call graph for this function:



#### 4.9.2.16 void SBML\_score\_mean ( pScore result, int n )

Mean quantities for score.

##### Author

Amine Ghozlane

##### Parameters

<i>result</i>	Struct <a href="#">Score</a>
<i>n</i>	Number of simulation step

Definition at line 591 of file simulation.c.

References Score::quantite, and Score::taille.

Referenced by SBML\_compute\_simulation\_mean().

#### 4.9.2.17 void SBML\_setReactions ( Model\_t \* mod, pEspecies molecules, pScore result, double \* reactions\_ratio, int nbReactions, int nbEspecies )

Alloc memory and initialize the struct [Especies](#).

##### Author

Amine Ghozlane

##### Parameters

<i>mod</i>	Model of the SBML file
<i>molecules</i>	Struct <a href="#">Especies</a>
<i>result</i>	Struct <a href="#">Score</a>
<i>reactions_ratio</i>	List of computed reaction ratio

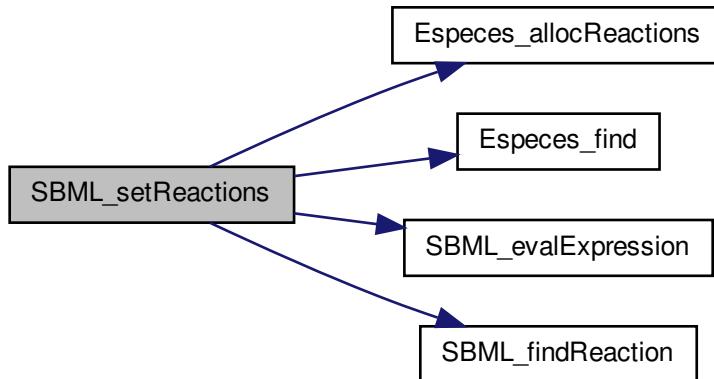
<i>nbReactions</i>	Number of reaction
<i>nbEspecies</i>	Number of molecules

Definition at line 105 of file simulation.c.

References `Especies_allocReactions()`, `Especies_find()`, `Score::nb_reaction`, `Score::reaction`, `SBML_evalExpression()`, and `SBML_findReaction()`.

Referenced by `SBML_compute_simulation()`, and `SBML_compute_simulation_mean()`.

Here is the call graph for this function:



**4.9.2.18** `int SBML_simulate ( Model_t * mod, pEspecies molecules, const gsl_rng * r, pTestReaction T, char ** banned, int nbBanned, int nbEspecies, int ref )`

Simulate one step of petri net.

#### Author

Amine Ghozlane

#### Parameters

<i>mod</i>	Model of the SBML file
<i>molecules</i>	Struct <code>Especies</code>
<i>r</i>	Random number generator
<i>T</i>	Struct <code>TestReaction</code> gives data on reaction
<i>banned</i>	List of banned compound
<i>nbBanned</i>	Number of banned compound

<i>nbEspecies</i>	Number of molecules
<i>ref</i>	Number reference of one molecule

**Returns**

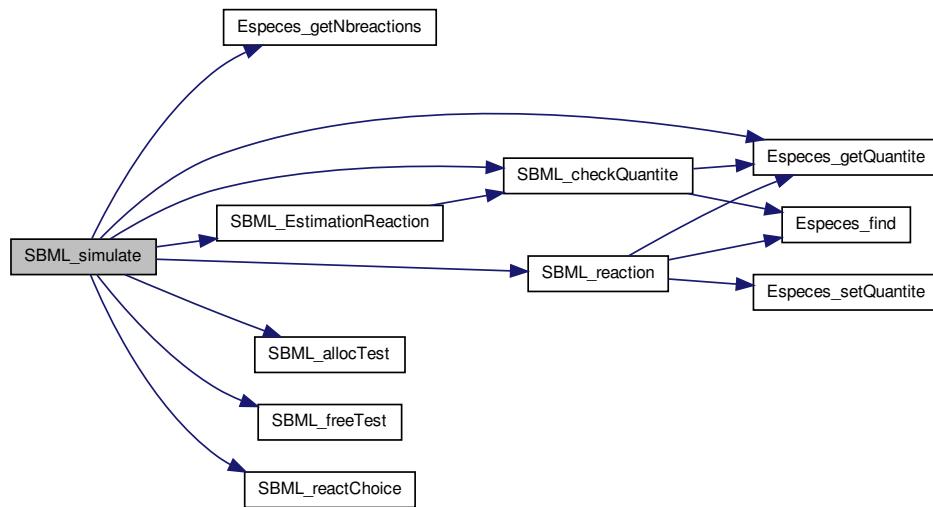
Condition of stop/pursue

Definition at line 356 of file simulation.c.

References Especies\_getNbReactions(), Especies\_getQuantite(), Reaction::link, SBML\_allocTest(), SBML\_checkQuantite(), SBML\_EstimationReaction(), SBML\_freeTest(), SBML\_reactChoice(), SBML\_reaction(), and Especies::system.

Referenced by SBML\_compute\_simulation(), and SBML\_compute\_simulation\_mean().

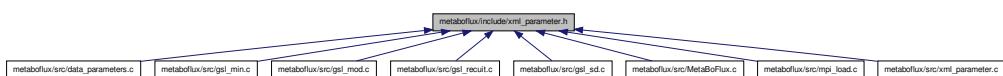
Here is the call graph for this function:



## 4.10 metaboflux/include/xml\_parameter.h File Reference

Xml reader for parametre.xml.

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



## Data Structures

- struct `xmlConfig_t`  
*Structure containing all information from parameter.xml.*

## Functions

- `xmlConfig_t * Xml_loadConfig (char *)`  
*Initialize and load the parameter.xml.*
- `void Xml_freeConfig (xmlConfig_t *)`  
*Free the Struct `xmlConfig_t`.*
- `int Xml_getNumber (xmlConfig_t *, const char *)`  
*Get integer value.*
- `double Xml_getDoubleNumber (xmlConfig_t *, const char *)`  
*Get double value.*
- `char * Xml_getString (xmlConfig_t *)`  
*Get name value.*
- `int Xml_getNbSimulations (xmlConfig_t *)`  
*Get number of simulation.*
- `int Xml_getNbTriesMod (xmlConfig_t *)`  
*Get number of tries for modelling.*
- `int Xml_getNbTriesSa (xmlConfig_t *)`  
*Get number of tries for the simulated annealing and minimization.*
- `int Xml_getNbIters (xmlConfig_t *)`  
*Get number of iteration.*
- `int Xml_getNbGroup (xmlConfig_t *)`  
*Get number of group.*
- `double Xml_getStepSize (xmlConfig_t *)`  
*Get the step size.*
- `double Xml_getBoltzmann (xmlConfig_t *)`  
*Get the Boltzmann value.*
- `double Xml_getTinitial (xmlConfig_t *)`  
*Get the initial temperature.*
- `double Xml_getMuT (xmlConfig_t *)`  
*Get the variation of temperature.*
- `double Xml_getTmin (xmlConfig_t *)`  
*Get the minimum temperature.*
- `int Xml_getCount (xmlConfig_t *, const char *)`  
*Get count.*
- `int Xml_getNbCouples (xmlConfig_t *)`  
*Get the number of couples.*
- `int Xml_getNbParameters (xmlConfig_t *)`  
*Get the number of parameters.*

- int `Xml_getNbReactioninNoeud (xmlConfig_t *, int)`  
*Count the number of reaction in one node.*
- int `Xml_getNbReaction (xmlConfig_t *)`  
*Get the number of reactions.*
- int `Xml_getNbSpecies (xmlConfig_t *)`  
*Get the number of species.*
- int `Xml_getNbEquations (xmlConfig_t *)`  
*Get the number of equations.*
- int `Xml_getNbBannedSpecies (xmlConfig_t *)`  
*Get the number of banned species.*
- char \*\* `Xml_getReactionsNamesinNoeud (xmlConfig_t *, int)`  
*Get list of reactions.*
- char \*\* `Xml_getReactionsNames (xmlConfig_t *)`  
*Get name of reactions.*
- char \*\* `Xml_getBanned (xmlConfig_t *conf)`  
*Get list of banned species.*
- int `Xml_getSpeciesFinalAmount (xmlConfig_t *, char *)`  
*Get the final amount of one species.*
- int \* `Xml_getallSpeciesFinalAmount (xmlConfig_t *, char **, int)`  
*Get Table of species amount.*
- int `Xml_getSpeciesWeight (xmlConfig_t *, char *)`  
*Get Species weight.*
- int \* `Xml_getallSpeciesWeight (xmlConfig_t *, char **, int)`  
*Get list of species weight.*
- char \*\* `Xml_allocEquation (char **, int)`  
*Allocate memory for the list of equation.*
- int `Xml_getKineticLawNodes (xmlConfig_t *, int)`  
*Get the kinetic law for one node.*
- char \*\*\* `Xml_getKineticLaw (xmlConfig_t *, int, int)`  
*Get list of kinetic laws.*

#### 4.10.1 Detailed Description

Xml reader for parametre.xml. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflux/>)  
 Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

{Amine Ghozlane}

**Version**

2.0

**Date**

15 janvier 2010

Definition in file [xml\\_parameter.h](#).

## 4.10.2 Function Documentation

### 4.10.2.1 `char** Xml_allocEquation ( char ** equation, int nb_noeud )`

Allocate memory for the list of equation.

**Author**

Amine Ghozlane

**Parameters**

<i>equation</i>	list of equation
<i>nb_noeud</i>	Number of the node

**Returns**

List of reactions

Definition at line 789 of file [xml\\_parameter.c](#).Referenced by [Xml\\_getKineticLaw\(\)](#).

### 4.10.2.2 `void Xml_freeConfig ( xmlConfig_t * conf )`

Free the Struct [xmlConfig\\_t](#).**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

Definition at line 97 of file [xml\\_parameter.c](#).References [xmlConfig\\_t::ctxt](#), [xmlConfig\\_t::doc](#), and [xmlConfig\\_t::fichier](#).

Referenced by Data\_desallocation(), and Xml\_loadConfig().

```
4.10.2.3 int* Xml_getallSpeciesFinalAmount ( xmlConfig_t * conf, char ** species, int taille
)
```

Get Table of species amount.

#### Author

Amine Ghozlane

#### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>species</i>	Table of species
<i>taille</i>	Number of species

#### Returns

Table of species amount

Definition at line 708 of file [xml\\_parameter.c](#).

References [Xml\\_getSpeciesFinalAmount\(\)](#).

Referenced by [Data\\_allocScore\(\)](#).

Here is the call graph for this function:



```
4.10.2.4 int* Xml_getallSpeciesWeight ( xmlConfig_t * conf, char ** species, int taille )
```

Get list of species weight.

#### Author

Amine Ghozlane

#### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>species</i>	Table of species
<i>taille</i>	Number of species

**Returns**

List of species weight

Definition at line 770 of file xml\_parameter.c.

References Xml\_getSpeciesWeight().

Referenced by Data\_allocScore().

Here is the call graph for this function:

**4.10.2.5 char\*\* Xml\_getBanned ( xmlConfig\_t \* conf )**

Get list of banned species.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

List of banned species

Definition at line 624 of file xml\_parameter.c.

References xmlConfig\_t::ctxt, and Xml\_getNbBannedSpecies().

Referenced by Data\_allocParameters().

Here is the call graph for this function:



#### 4.10.2.6 double Xml\_getBoltzmann ( *xmlConfig\_t* \* *conf* )

Get the Boltzmann value.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

##### Returns

Boltzmann value

Definition at line 305 of file `xml_parameter.c`.

References `Xml_getDoubleNumber()`.

Referenced by `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:



#### 4.10.2.7 int Xml\_getCount ( *xmlConfig\_t* \* *conf*, const char \* *requete* )

Get count.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>requete</i>	Xpath query

**Returns**

Count

Definition at line 358 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Xml_getNbBannedSpecies()`, `Xml_getNbCouples()`, `Xml_getNbEquations()`, `Xml_getNbParameters()`, `Xml_getNbReaction()`, `Xml_getNbReactioninNoeud()`, and `Xml_getNbSpecies()`.

**4.10.2.8 double Xml\_getDoubleNumber ( `xmlConfig_t * conf, const char * requete` )**

Get double value.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>requete</i>	Xpath query

**Returns**

Read double value in the xml

Definition at line 153 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Xml_getBoltzmann()`, `Xml_getMuT()`, `Xml_getStepSize()`, `Xml_getTinitial()`, and `Xml_getTmin()`.

**4.10.2.9 char\*\*\* Xml\_getKineticLaw ( `xmlConfig_t * conf, int num_equation, int nb_noeud` )**

Get list of kinetic laws.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>num_-equation</i>	Number of equation
<i>nb_noeud</i>	Number nodes

**Returns**

List of kinetic laws

Definition at line 854 of file xml\_parameter.c.

References [xmlConfig\\_t::ctxt](#), and [Xml\\_allocEquation\(\)](#).

Referenced by [Data\\_allocEquations\(\)](#).

Here is the call graph for this function:



#### 4.10.2.10 int Xml\_getKineticLawNodes ( *xmlConfig\_t \* conf*, *int num\_equation* )

Get the kinetic law for one node.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>num_-equation</i>	Number of the node

**Returns**

Ninetic law for one node

Definition at line 813 of file xml\_parameter.c.

References [xmlConfig\\_t::ctxt](#).

Referenced by [Data\\_allocEquations\(\)](#).

**4.10.2.11 double Xml\_getMuT( xmlConfig\_t \* conf )**

Get the variation of temperature.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

Variation of temperature

Definition at line 331 of file `xml_parameter.c`.

References `Xml_getDoubleNumber()`.

Referenced by `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:

**4.10.2.12 int Xml\_getNbBannedSpecies( xmlConfig\_t \* conf )**

Get the number of banned species.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

Number of banned species

Definition at line 486 of file `xml_parameter.c`.

References `Xml_getCount()`.

Referenced by `Data_allocParameters()`, and `Xml_getBanned()`.

Here is the call graph for this function:



#### 4.10.2.13 int `Xml_getNbCouples ( xmlConfig_t * conf )`

Get the number of couples.

##### Author

Amine Ghozlane

##### Parameters

<code>conf</code>	Struct <a href="#">xmlConfig_t</a>
-------------------	------------------------------------

##### Returns

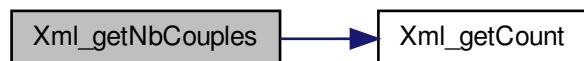
Number of couples

Definition at line 396 of file `xml_parameter.c`.

References `Xml_getCount()`.

Referenced by `Data_allocParameters()`, and `Xml_getReactionsNames()`.

Here is the call graph for this function:



**4.10.2.14 int Xml\_getNbEquations ( xmlConfig\_t \* conf )**

Get the number of equations.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

Number of equations

Definition at line 473 of file `xml_parameter.c`.

References `Xml_getCount()`.

Referenced by `Data_allocEquations()`.

Here is the call graph for this function:

**4.10.2.15 int Xml\_getNbGroup ( xmlConfig\_t \* conf )**

Get number of group.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

Number of iteration

Definition at line 279 of file `xml_parameter.c`.

References `Xml_getNumber()`.

Referenced by `Mpi_master()`.

Here is the call graph for this function:



#### 4.10.2.16 int `Xml_getNbIter` ( `xmlConfig_t * conf` )

Get number of iteration.

##### Author

Amine Ghozlane

##### Parameters

<code>conf</code>	Struct <a href="#">xmlConfig_t</a>
-------------------	------------------------------------

##### Returns

Number of iteration

Definition at line 266 of file `xml_parameter.c`.

References `Xml_getNumber()`.

Referenced by `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:



**4.10.2.17 int Xml\_getNbParameters ( xmlConfig\_t \* *conf* )**

Get the number of parameters.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

Number of parameters

Definition at line 409 of file `xml_parameter.c`.

References `Xml_getCount()`.

Referenced by `Data_allocParameters()`.

Here is the call graph for this function:

**4.10.2.18 int Xml\_getNbReaction ( xmlConfig\_t \* *conf* )**

Get the number of reactions.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

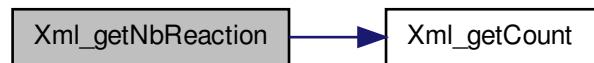
Number of reactions

Definition at line 447 of file `xml_parameter.c`.

References `Xml_getCount()`.

Referenced by Data\_allocScore(), and Xml\_getReactionsNames().

Here is the call graph for this function:



#### 4.10.2.19 int Xml\_getNbReactioninNoeud ( xmlConfig\_t \* conf, int noeud )

Count the number of reaction in one node.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>noeud</i>	Number of the node

##### Returns

Number of reaction in one node

Definition at line 423 of file [xml\\_parameter.c](#).

References [Xml\\_getCount\(\)](#).

Referenced by [Data\\_allocParameters\(\)](#), and [Xml\\_getReactionsNamesinNoeud\(\)](#).

Here is the call graph for this function:



**4.10.2.20 int Xml\_getNbSimulations ( xmlConfig\_t \* conf )**

Get number of simulation.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

Number of simulation

Definition at line 227 of file `xml_parameter.c`.

References `Xml_getNumber()`.

Referenced by `Mpi_master()`.

Here is the call graph for this function:

**4.10.2.21 int Xml\_getNbSpecies ( xmlConfig\_t \* conf )**

Get the number of species.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

Number of species

Definition at line 460 of file `xml_parameter.c`.

References `Xml_getCount()`.

Referenced by Data\_allocScore(), Mpi\_slave(), and Mpi\_writer().

Here is the call graph for this function:



#### 4.10.2.22 int Xml\_getNbTriesMod ( *xmlConfig\_t* \* *conf* )

Get number of tries for modelling.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

##### Returns

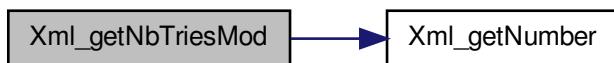
Number of tries

Definition at line 240 of file xml\_parameter.c.

References Xml\_getNumber().

Referenced by Data\_allocParameters(), Mod\_compute\_modeling(), Mpi\_sizeResultTab(), and Sd\_compute\_standard\_deviation().

Here is the call graph for this function:



#### 4.10.2.23 int Xml\_getNbTriesSa ( *xmlConfig\_t* \* *conf* )

Get number of tries for the simulated annealing and minimization.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

##### Returns

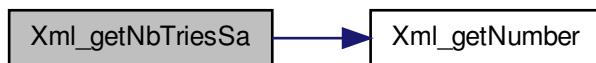
Number of tries

Definition at line 253 of file `xml_parameter.c`.

References `Xml_getNumber()`.

Referenced by `Data_allocParameters()`, `Min_my_f()`, `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:



#### 4.10.2.24 int Xml\_getNumber ( *xmlConfig\_t* \* *conf*, const char \* *requete* )

Get integer value.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>requete</i>	Xpath query

##### Returns

Read integer value in the xml

Definition at line 115 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Xml_getNbGroup()`, `Xml_getNbIter()`, `Xml_getNbSimulations()`, `Xml_getNbTriesMod()`, and `Xml_getNbTriesSa()`.

#### 4.10.2.25 `char** Xml_getReactionsNames ( xmlConfig_t * conf )`

Get name of reactions.

##### Author

Amine Ghozlane

##### Parameters

<code>conf</code>	Struct <a href="#">xmlConfig_t</a>
-------------------	------------------------------------

##### Returns

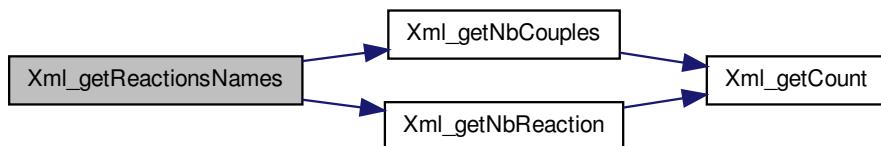
Name of reactions

Definition at line 541 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`, `Xml_getNbCouples()`, and `Xml_getNbReaction()`.

Referenced by `Data_allocScore()`.

Here is the call graph for this function:



#### 4.10.2.26 `char** Xml_getReactionsNamesinNoeud ( xmlConfig_t * conf, int numero )`

Get list of reactions.

Get list of species.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>numero</i>	Number of the node

**Returns**

List of reactions

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

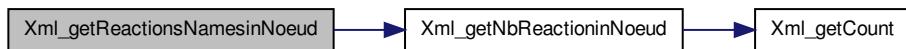
**Returns**

List of species

Definition at line 500 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`, and `Xml_getNbReactioninNoeud()`.

Here is the call graph for this function:



#### 4.10.2.27 int Xml\_getSpeciesFinalAmount ( `xmlConfig_t * conf, char * species` )

Get the final amount of one species.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>species</i>	Name of one species

**Returns**

Final amount of one species

Definition at line 665 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Xml_getallSpeciesFinalAmount()`.

4.10.2.28 `int Xml_getSpeciesWeight ( xmlConfig_t * conf, char * species )`

Get Species weight.

#### Author

Amine Ghozlane

#### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>species</i>	Name of one species

#### Returns

Species weight

Definition at line 727 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Xml_getallSpeciesWeight()`.

4.10.2.29 `double Xml_getStepSize ( xmlConfig_t * conf )`

Get the step size.

#### Author

Amine Ghozlane

#### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

#### Returns

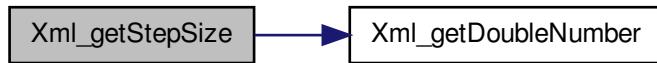
Step size

Definition at line 292 of file `xml_parameter.c`.

References `Xml_getDoubleNumber()`.

Referenced by `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:



#### 4.10.2.30 `char* XmlGetString ( xmlConfig_t * conf )`

Get name value.

##### Author

Amine Ghozlane

##### Parameters

<code>conf</code>	Struct <a href="#">xmlConfig_t</a>
-------------------	------------------------------------

##### Returns

Read name value in the xml

Definition at line 190 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Recuit_redirectionFlux()`.

#### 4.10.2.31 `double Xml_getTinitial ( xmlConfig_t * conf )`

Get the initial temperature.

##### Author

Amine Ghozlane

##### Parameters

<code>conf</code>	Struct <a href="#">xmlConfig_t</a>
-------------------	------------------------------------

##### Returns

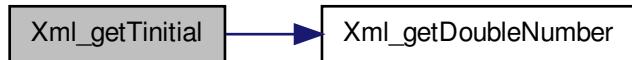
Initial temperature

Definition at line 318 of file `xml_parameter.c`.

References `Xml_getDoubleNumber()`.

Referenced by `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:



#### 4.10.2.32 double `Xml_getTmin ( xmlConfig_t * conf )`

Get the minimum temperature.

##### Author

Amine Ghozlane

##### Parameters

<code>conf</code>	Struct <a href="#">xmlConfig_t</a>
-------------------	------------------------------------

##### Returns

Minimum temperature

Definition at line 344 of file `xml_parameter.c`.

References `Xml_getDoubleNumber()`.

Referenced by `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:



#### 4.10.2.33 `xmlConfig_t* Xml_loadConfig( char * fichier )`

Initialize and load the parameter.xml.

##### Author

Amine Ghozlane

##### Parameters

<code>fichier</code>	Xml file name
----------------------	---------------

##### Returns

Struct [xmlConfig\\_t](#)

Definition at line 42 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`, `xmlConfig_t::doc`, `xmlConfig_t::fichier`, `xmlConfig_t::racine`, and `Xml_freeConfig()`.

Referenced by `Data_initXML()`.

Here is the call graph for this function:



## 4.11 metaboflux/src/data\_parameters.c File Reference

Load data parameters.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <time.h>
#include <math.h>
#include <sys/types.h>
#include <unistd.h>
```

```
#include <gsl/gsl blas.h>
#include <gsl/gsl_multimin.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_siman.h>
#include <gsl/gsl_ieee_utils.h>
#include <sbml/SBMLTypes.h>
#include <libxml/parser.h>
#include <libxml>xpath.h>
#include "xml_parameter.h"
#include "especies.h"
#include "equations.h"
#include "simulation.h"
#include "data_parameters.h"
```

Include dependency graph for data\_parameters.c:



## Functions

- `xmlConfig_t * Data_initXML (char *xml_file)`  
*Load parameter file.*
- `Model_t * Data_initSBML (char *sbml_file)`  
*Load SBML file.*
- `gsl_rng * Data_rngAlloc (gsl_rng *r)`  
*Allocation of the random number generator.*
- `Equations ** Data_equationsAlloc (pListParameters current, Equations **pile)`  
*Allocate the struct `Equations`.*
- `void Data_equationsInit (pListParameters current, Equations **pile)`  
*Initialize equations.*
- `void Data_allocEquations (pListParameters a)`  
*Allocate the different elements related to the struct `Equations`.*
- `void Data_allocScore (pScore out, xmlConfig_t *conf, Model_t *model)`  
*Copy the partial table x in the full table y.*
- `pScore Data_scoreAlloc (pListParameters a)`  
*Allocation of the struct `Score`.*
- `void Data_scoreInit (pScore out)`

- Initialize the struct `Score`.*
- void `Data_scoreFree (pScore out)`
  - Free the struct `Score`.*
- int `Data_copieTab (pSimParameters simulated, double *x, int a, int debut, int fin)`
  - Copy the partial table x in the full table y.*
- void `Data_updateTab (pListParameters current, pSimParameters simulated, double *x)`
  - Copy table.*
- void `Data_allocSimParameters (double *y, pListParameters a)`
  - Allocation of tables needed for simulations.*
- void `Data_allocParameters (pListParameters a)`
  - Copy the partial table x in the full table y.*
- void `Data_parameters (pListParameters a, char **files_path)`
  - Record parameters of the simulation.*
- void `Data_desallocation (pListParameters a)`
  - Free memory allocated to the various objects of the program.*
- void `Data_simParameters (pListParameters a, pSimParameters sim, char *out, char *texte, int number, int debug)`
  - Recording parameters specific to each simulation.*
- void `Data_desallocSim (pSimParameters sim)`
  - Free struct parameters specific to each simulation.*

#### 4.11.1 Detailed Description

Load data parameters. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metabofl>)  
 Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

##### Author

{Amine Ghozlane}

##### Version

2.0

##### Date

10 novembre 2010

Definition in file [data\\_parameters.c](#).

#### 4.11.2 Function Documentation

##### 4.11.2.1 void Data\_allocEquations ( pListParameters a )

Allocate the different elements related to the struct [Equations](#).

###### Author

Amine Ghozlane

###### Parameters

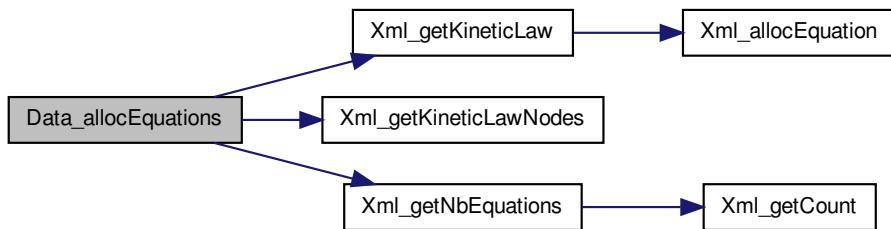
<i>a</i>	Global parameters : struct <a href="#">ListParameters</a>
----------	---

Definition at line 202 of file [data\\_parameters.c](#).

References [ListParameters::conf](#), [ListParameters::equation](#), [ListParameters::nb\\_equations](#), [ListParameters::noeud](#), [Xml\\_getKineticLaw\(\)](#), [Xml\\_getKineticLawNodes\(\)](#), and [Xml\\_getNbEquations\(\)](#).

Referenced by [Data\\_parameters\(\)](#).

Here is the call graph for this function:



##### 4.11.2.2 void Data\_allocParameters ( pListParameters a )

Copy the partial table x in the full table y.

###### Author

Amine Ghozlane

###### Parameters

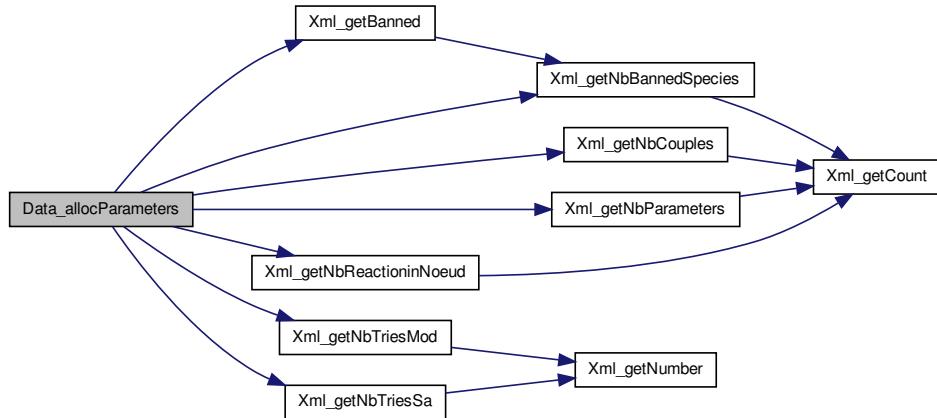
a	Global parameters : struct <a href="#">ListParameters</a>
---	---

Definition at line 408 of file `data_parameters.c`.

References `ListParameters::banned`, `ListParameters::conf`, `ListParameters::interest_parameters`, `ListParameters::nb_banned`, `ListParameters::nb_couples`, `ListParameters::nb_parameters`, `ListParameters::nb_triesMod`, `ListParameters::nb_triesSa`, `ListParameters::parameters`, `Xml_getBanned()`, `Xml_getNbBannedSpecies()`, `Xml_getNbCouples()`, `Xml_getNbParameters()`, `Xml_getNbReactioninNoeud()`, `Xml_getNbTriesMod()`, and `Xml_getNbTriesSa()`.

Referenced by `Data_parameters()`.

Here is the call graph for this function:



#### 4.11.2.3 void Data\_allocScore ( pScore *out*, xmlConfig\_t \* *conf*, Model\_t \* *model* )

Copy the partial table *x* in the full table *y*.

##### Author

Amine Ghozlane

##### Parameters

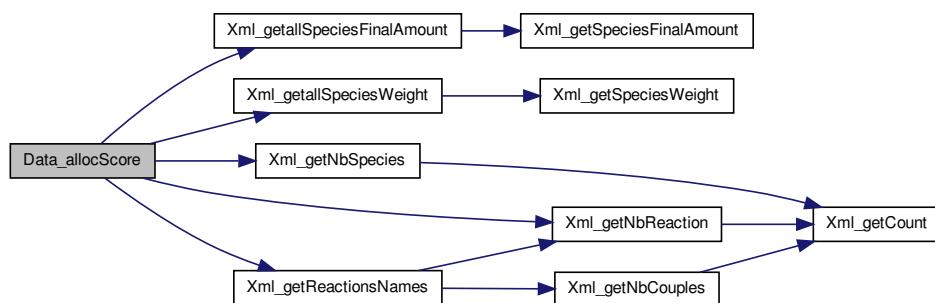
<i>out</i>	Struct <a href="#">Score</a>
<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>model</i>	Model of the SBML file

Definition at line 238 of file `data_parameters.c`.

References Score::name, Score::nb\_reaction, Score::nb\_species, Score::quantite, Score::reaction, Score::species, Score::species\_amount, Score::species\_weight, Score::taille, Score::tailleReactions, Score::tailleSpecies, Xml\_getallSpeciesFinalAmount(), Xml\_getallSpeciesWeight(), Xml\_getNbReaction(), Xml\_getNbSpecies(), and Xml\_getReactionsNames().

Referenced by Data\_scoreAlloc(), and Data\_simParameters().

Here is the call graph for this function:



#### 4.11.2.4 void Data\_allocSimParameters ( double \* y, pListParameters a )

#### Allocation of tables needed for simulations.

```
void Data allocSimParameters(double *y, pListParameters a)
```

## Author

Amine Ghozlane

## Parameters

<i>y</i>	table of reaction parameters
<i>a</i>	Global parameters : struct <a href="#">ListParameters</a>

Definition at line 391 of file data\_parameters.c.

References ListParameters::nb\_parameters.

Referenced by Data\_simParameters().

4.11.2.5 int Data\_copieTab ( pSimParameters simulated, double \* x, int a, int debut, int fin )

Copy the partial table x in the full table y.

**Author**

Amine Ghozlane

**Parameters**

<i>simulated</i>	Simulation parameters : struct <a href="#">SimParameters</a>
<i>x</i>	Short table of reaction parameters
<i>a</i>	Line
<i>debut</i>	Beginning
<i>fin</i>	End

**Returns**

Number of copied element

Definition at line 347 of file `data_parameters.c`.

References `SimParameters::y`.

Referenced by `Data_updateTab()`.

**4.11.2.6 void Data\_desallocation ( [pListParameters](#) a )**

Free memory allocated to the various objects of the program.

**Author**

Amine Ghozlane

**Parameters**

<i>a</i>	Global parameters : struct <a href="#">ListParameters</a>
----------	---

Definition at line 463 of file `data_parameters.c`.

References `ListParameters::banned`, `ListParameters::conf`, `ListParameters::equation`, `ListParameters::model`, `ListParameters::nb_equations`, `ListParameters::noeud`, `ListParameters::parameters`, and `Xml_freeConfig()`.

Referenced by `compute_mpi()`.

Here is the call graph for this function:



**4.11.2.7 void Data\_desallocSim ( pSimParameters sim )**

Free struct parameters specific to each simulation.

**Author**

Amine Ghozlane

**Parameters**

<i>sim</i>	Simulation parameters : struct <a href="#">SimParameters</a>
------------	--

Definition at line 549 of file data\_parameters.c.

References SimParameters::debugFile, Score::name, Score::nb\_reaction, SimParameters::out, Score::quantite, SimParameters::r, Score::reaction, Score::species, Score::species\_amount, Score::species\_weight, Score::taille, and SimParameters::y.

Referenced by Min\_compute\_minimization(), Mod\_compute\_modeling(), Recuit\_compute\_recuit(), and Sd\_compute\_standard\_deviation().

**4.11.2.8 Equations \*\* Data\_equationsAlloc ( pListParameters current, Equations \*\* pile )**

Allocate the struct [Equations](#).

**Author**

Amine Ghozlane

**Parameters**

<i>current</i>	Current parameters : struct <a href="#">ListParameters</a>
<i>pile</i>	Pile des equations : struct <a href="#">Equations</a>

**Returns**

Allocated struct [Equations](#)

Definition at line 169 of file data\_parameters.c.

References ListParameters::nb\_equations.

Referenced by Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

**4.11.2.9 void Data\_equationsInit ( pListParameters current, Equations \*\* pile )**

Initialize equations.

**Author**

Amine Ghozlane

**Parameters**

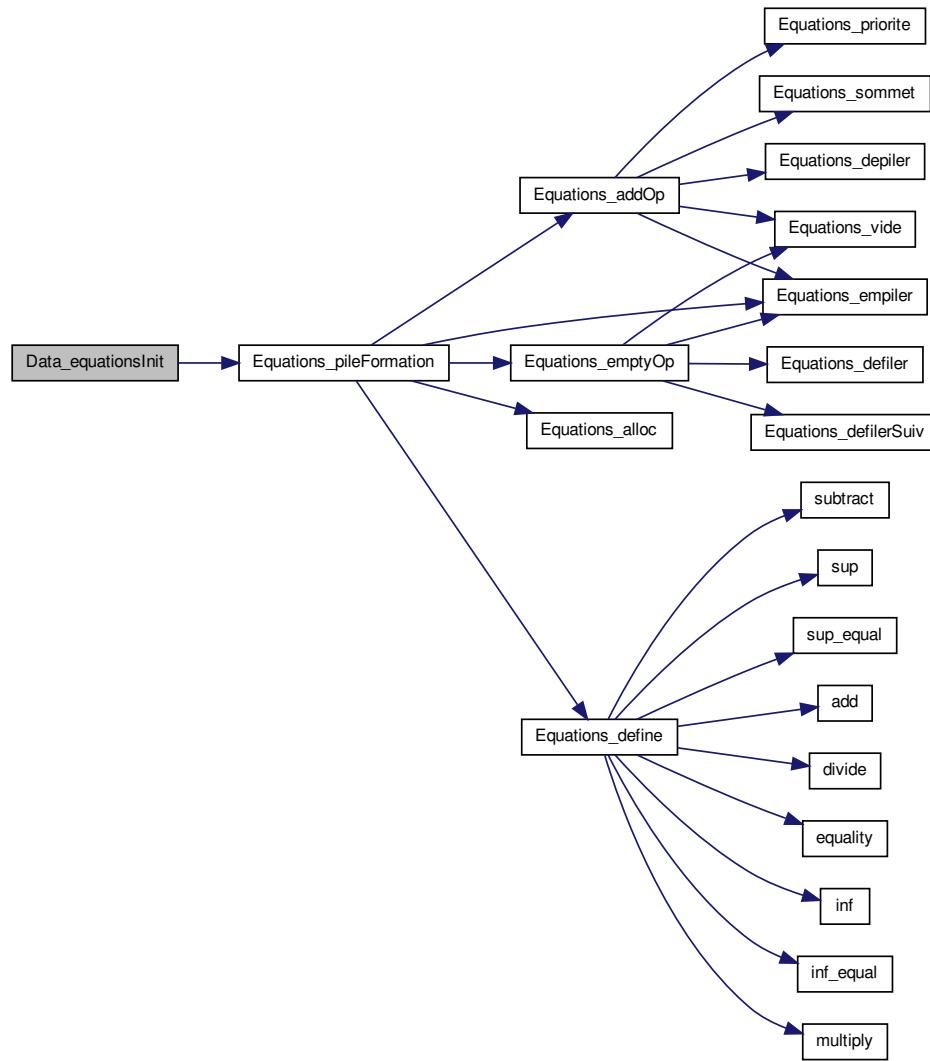
<i>current</i>	Current parameters : struct <a href="#">ListParameters</a>
<i>pile</i>	Pile des equations : struct <a href="#">Equations</a>

Definition at line 184 of file `data_parameters.c`.

References `ListParameters::equation`, `Equations_pileFormation()`, `ListParameters::nb_equations`, and `ListParameters::noeud`.

Referenced by `Min_my_f()`, `Mod_compute_modeling()`, `Recuit_energyFunction()`, and `Sd_compute_simulation()`.

Here is the call graph for this function:



#### 4.11.2.10 Model\_t \* Data\_initSBML ( char \* sbml\_file )

Load SBML file.

## Author

Amine Ghozlane

**Parameters**

<i>sbml_file</i>	Name of the SBML file
------------------	-----------------------

**Returns**

Model of the SBML file

Definition at line 81 of file data\_parameters.c.

Referenced by Data\_parameters().

**4.11.2.11 `xmlConfig_t * Data_initXML ( char * xml_file )`**

Load parameter file.

**Author**

Amine Ghozlane

**Parameters**

<i>xml_file</i>	Name of the parameter file
-----------------	----------------------------

**Returns**

Model of the parameter file =:Struct [xmlConfig\\_t](#)

Definition at line 58 of file data\_parameters.c.

References Xml\_loadConfig().

Referenced by Data\_parameters().

Here is the call graph for this function:

**4.11.2.12 `void Data_parameters ( pListParameters a, char ** files_path )`**

Record parameters of the simulation.

**Author**

Amine Ghozlane

## Parameters

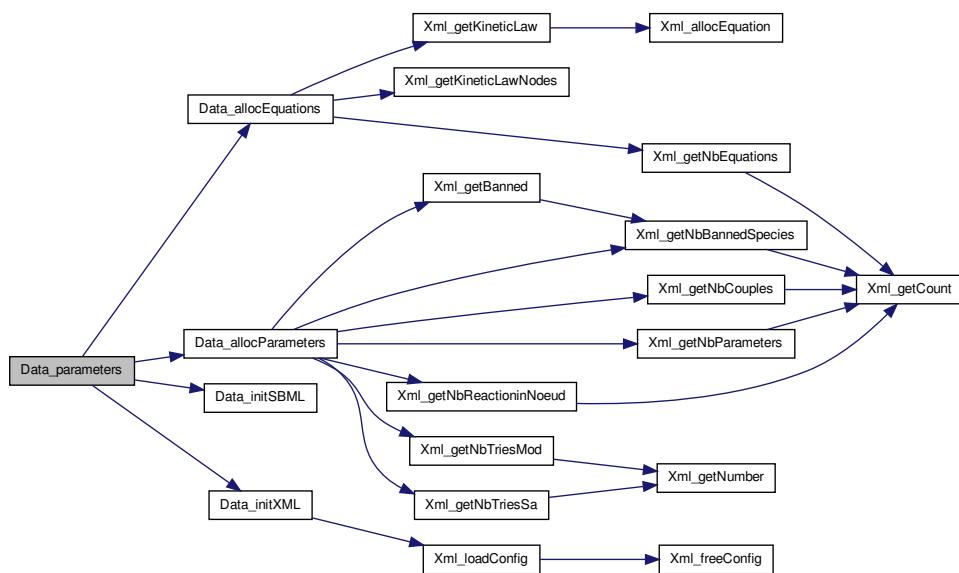
<i>a</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>files_path</i>	List of paths

Definition at line 442 of file data\_parameters.c.

References ListParameters::conf, Data\_allocEquations(), Data\_allocParameters(), Data\_initSBML(), Data\_initXML(), and ListParameters::model.

Referenced by compute\_mpi().

Here is the call graph for this function:



**4.11.2.13 `gsl_rng * Data_rngAlloc ( gsl_rng * r )`**

Allocation of the random number generator.

## Author

Amine Ghozlane

## Parameters

*r* Random number generator

## Returns

## Random number generator

Definition at line 135 of file data\_parameters.c.

Referenced by Data\_simParameters().

#### 4.11.2.14 pScore Data\_scoreAlloc ( pListParameters a )

Allocation of the struct [Score](#).

##### Author

Amine Ghozlane

##### Parameters

a	Global parameters : struct <a href="#">ListParameters</a>
---	---

##### Returns

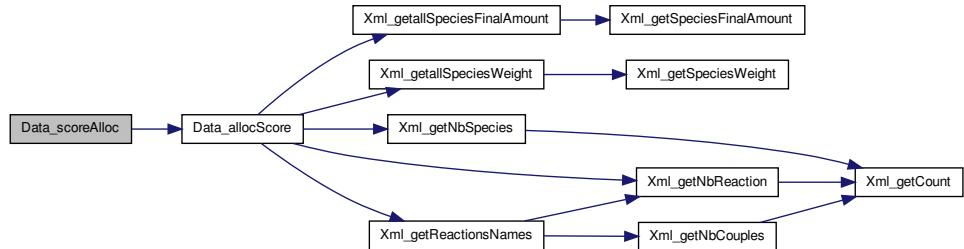
Allocated struct [Score](#)

Definition at line 274 of file data\_parameters.c.

References ListParameters::conf, Data\_allocScore(), and ListParameters::model.

Referenced by Min\_compute\_minimization(), Mod\_compute\_modeling(), and Recuit\_compute\_recuit().

Here is the call graph for this function:



#### 4.11.2.15 void Data\_scoreFree ( pScore out )

Free the struct [Score](#).

##### Author

Amine Ghozlane

##### Parameters

<i>out</i>	Struct <a href="#">score</a>
------------	------------------------------

Definition at line 307 of file data\_parameters.c.

References [Score::name](#), [Score::nb\\_reaction](#), [Score::quantite](#), [Score::reaction](#), [Score::species\\_amount](#), [Score::species\\_weight](#), and [Score::taille](#).

Referenced by [Min\\_compute\\_minimization\(\)](#), [Mod\\_compute\\_modeling\(\)](#), and [Recuit\\_compute\\_recuit\(\)](#).

#### 4.11.2.16 void Data\_scoreInit ( [pScore](#) *out* )

Initialize the struct [Score](#).

##### Author

Amine Ghozlane

##### Parameters

<i>out</i>	Empty struct <a href="#">Score</a>
------------	------------------------------------

Definition at line 290 of file data\_parameters.c.

References [Score::quantite](#), and [Score::taille](#).

Referenced by [Min\\_my\\_f\(\)](#), [Recuit\\_energyFunction\(\)](#), and [Sd\\_compute\\_simulation\(\)](#).

#### 4.11.2.17 void Data\_simParameters ( [pListParameters](#) *a*, [pSimParameters](#) *sim*, char \* *out*, char \* *texte*, int *number*, int *debug* )

Recording parameters specific to each simulation.

##### Author

Amine Ghozlane

##### Parameters

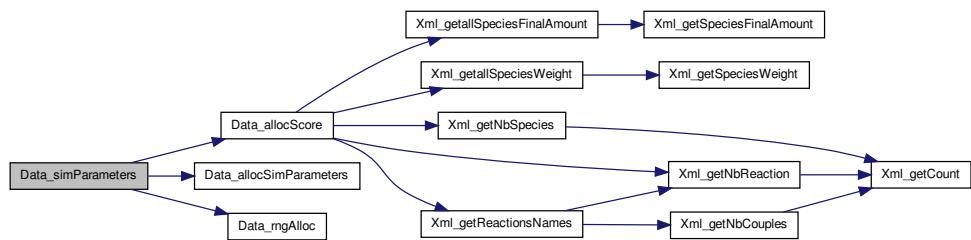
<i>a</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>sim</i>	Simulation parameters : struct <a href="#">SimParameters</a>
<i>out</i>	Output repertory
<i>texte</i>	File name
<i>number</i>	Number of the debug file
<i>debug</i>	Determine debug

Definition at line 506 of file data\_parameters.c.

References [ListParameters::conf](#), [Data\\_allocScore\(\)](#), [Data\\_allocSimParameters\(\)](#), [Data\\_rngAlloc\(\)](#), [SimParameters::debugFile](#), [ListParameters::model](#), [ListParameters::nb\\_parameters](#), [SimParameters::out](#), [SimParameters::pile](#), [SimParameters::r](#), and [SimParameters::y](#).

Referenced by Min\_compute\_minimization(), Mod\_compute\_modeling(), Recuit\_compute\_recuit(), and Sd\_compute\_standard\_deviation().

Here is the call graph for this function:



**4.11.2.18 void Data\_updateTab ( pListParameters *current*, pSimParameters *simulated*, double \* *x* )**

Copy table.

#### Author

Amine Ghozlane

#### Parameters

<i>current</i>	Current parameters : struct <a href="#">ListParameters</a>
<i>simulated</i>	Simulation parameters : struct <a href="#">SimParameters</a>
<i>x</i>	Short table of reaction parameters

Definition at line 371 of file `data_parameters.c`.

References `Data_copieTab()`, `ListParameters::nb_couples`, and `ListParameters::parameters`.

Referenced by `Min_my_f()`, `Recuit_energyFunction()`, and `Recuit_printPosition()`.

Here is the call graph for this function:

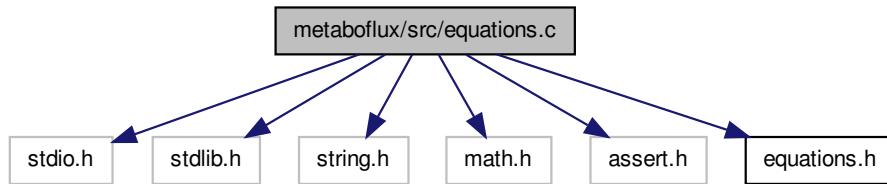


## 4.12 metaboflux/src/equations.c File Reference

Processes an equation in MathML format.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <assert.h>
#include "equations.h"
```

Include dependency graph for equations.c:



## Functions

- double **add** (double arg1, double arg2)  
*Add two numbers.*
- double **subtract** (double arg1, double arg2)  
*Subtraction two numbers.*
- double **multiply** (double arg1, double arg2)  
*Multiply two numbers.*
- double **divide** (double arg1, double arg2)  
*Divide two numbers.*
- double **equality** (double arg1, double arg2)  
*Test the equality of two numbers.*
- double **sup** (double arg1, double arg2)  
*Test the superiority between two numbers.*
- double **sup\_equal** (double arg1, double arg2)  
*Test the superiority or equality between two numbers.*
- double **inf** (double arg1, double arg2)  
*Test the inferiority between two numbers.*

- double `inf_equal` (double arg1, double arg2)  
*Test the inferiority or equality between two numbers.*
- `pEquations Equations_alloc` (void)  
*Alloc memory and initialize the struct `Equations`.*
- void `Equations_define` (`pEquations` new, char \*opérateur)  
*Identify the mathematical operator used.*
- int `Equations_vide` (`pEquations` liste)  
*Test if the struct `Equations` is empty.*
- `pEquations Equations_empiler` (`pEquations` liste, `pEquations` new)  
*Stack an element to the struct `Equations`.*
- `pEquations Equations_depiler` (`pEquations` liste)  
*Unstack the last element of the struct `Equations`.*
- `pEquations Equations_sommet` (`pEquations` liste)  
*Look for the last element of the struct `Equations`.*
- `pEquations Equations_defiler` (`pEquations` liste)  
*Look for the first element of the struct `Equations`.*
- `pEquations Equations_defilerSuiv` (`pEquations` liste)  
*Get the next element of the struct `Equations`.*
- int `Equations_priorite` (`pEquations` new)  
*Give the priority of the operator. Multiplication and Division have higher priority than addition or subtraction...*
- void `Equations_print` (`pEquations` liste)  
*Print all information on the element of the struct `Equations`.*
- `pEquations Equations_addOp` (`pEquations` result, `pEquations` op, `pEquations` new)  
*Build the list of the Struct `Equations` result used to compute the equation.*
- void `Equations_emptyOp` (`pEquations` result, `pEquations` op)  
*Empty the Struct `Equations` op for the struct Equation result.*
- `pEquations Equations_pileFormation` (char \*\*\*equation, int nb\_noeud)  
*Build pile.*
- double `Equations_extractData` (`pEquations` new, char \*\*name, double \*quantite, int taille)  
*Extract information on element of type Struct Equation.*
- double `Equations_findSpecies` (char \*species, char \*\*name, double \*quantite, int taille)  
*Look for the quantity of a molecule in the list.*
- double `Equations_resultat` (`pEquations` result, `pEquations` result1, `pEquations` eq, char \*\*name, double \*quantite, int taille)  
*Compute the result of two pile.*
- double `Equations_calcul` (`pEquations` liste, char \*\*name, double \*quantite, int taille)  
*Compute the score of the equation.*
- double `Equations_finalQuantite` (int file\_nb\_especes, char \*\*file\_species, int \*file\_amount, int \*file\_weight, char \*\*sim\_name, double \*sim\_quantite, int sim\_taille)

*Compute the score of the quantity which means the difference between what is expected to what is simulated.*

#### 4.12.1 Detailed Description

Processes an equation in MathML format. This file is part of Metaboflux (<http://www.cbib.u-bordeaux2.fr/metabolux/>)  
Copyright (C) 2010 Amine Ghazlane from LaBRI and University of Bordeaux 1

Metaboflux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Metaboflux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

##### Author

{Amine Ghazlane}

##### Version

2.0

##### Date

15 janvier 2010

Definition in file [equations.c](#).

#### 4.12.2 Function Documentation

##### 4.12.2.1 double add ( double arg1, double arg2 )

Add two numbers.

##### Author

Amine Ghazlane

##### Parameters

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

##### Returns

Result

Definition at line 43 of file equations.c.

Referenced by Equations\_define().

#### 4.12.2.2 double divide ( double *arg1*, double *arg2* )

Divide two numbers.

##### Author

Amine Ghozlane

##### Parameters

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

##### Returns

Result

Definition at line 82 of file equations.c.

Referenced by Equations\_define().

#### 4.12.2.3 double equality ( double *arg1*, double *arg2* )

Test the equality of two numbers.

##### Author

Amine Ghozlane

##### Parameters

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

##### Returns

Result

Definition at line 95 of file equations.c.

Referenced by Equations\_define().

#### 4.12.2.4 pEquations Equations\_addOp ( pEquations *result*, pEquations *op*, pEquations *new* )

Build the list of the Struct Equations result used to compute the equation.

**Author**

Amine Ghozlane

**Parameters**

<i>result</i>	Struct <a href="#">Equations</a> used for computation
<i>op</i>	Struct <a href="#">Equations</a> used to store the operator
<i>new</i>	One element of the Struct <a href="#">Equations</a>

**Returns**

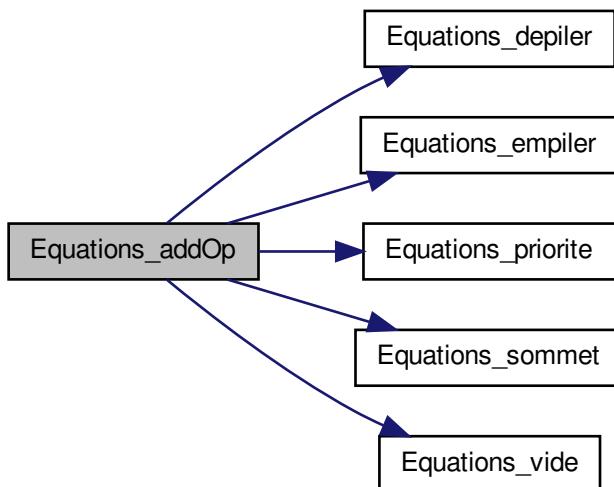
List of operator

Definition at line 383 of file equations.c.

References equal, [Equations\\_depiler\(\)](#), [Equations\\_empiler\(\)](#), [Equations\\_priorite\(\)](#), [Equations\\_sommet\(\)](#), [Equations\\_vide\(\)](#), inferior, inferior\_equal, superior, and superior\_equal.

Referenced by [Equations\\_pileFormation\(\)](#).

Here is the call graph for this function:

**4.12.2.5 pEquations Equations\_alloc ( void )**

Alloc memory and initialize the struct [Equations](#).

**Author**

Amine Ghozlane

**Returns**

Allocated struct [Equations](#)

Definition at line 158 of file equations.c.

Referenced by Equations\_calcul(), and Equations\_pileFormation().

**4.12.2.6 double Equations\_calcul ( pEquations liste, char \*\* name, double \* quantite, int taille )**

Compute the score of the equation.

**Author**

Amine Ghozlane

**Parameters**

<i>liste</i>	pile of type Struct <a href="#">Equations</a>
<i>name</i>	List of molecules
<i>quantite</i>	List of the quantity of the molecules
<i>taille</i>	Number of molecules in the list

**Returns**

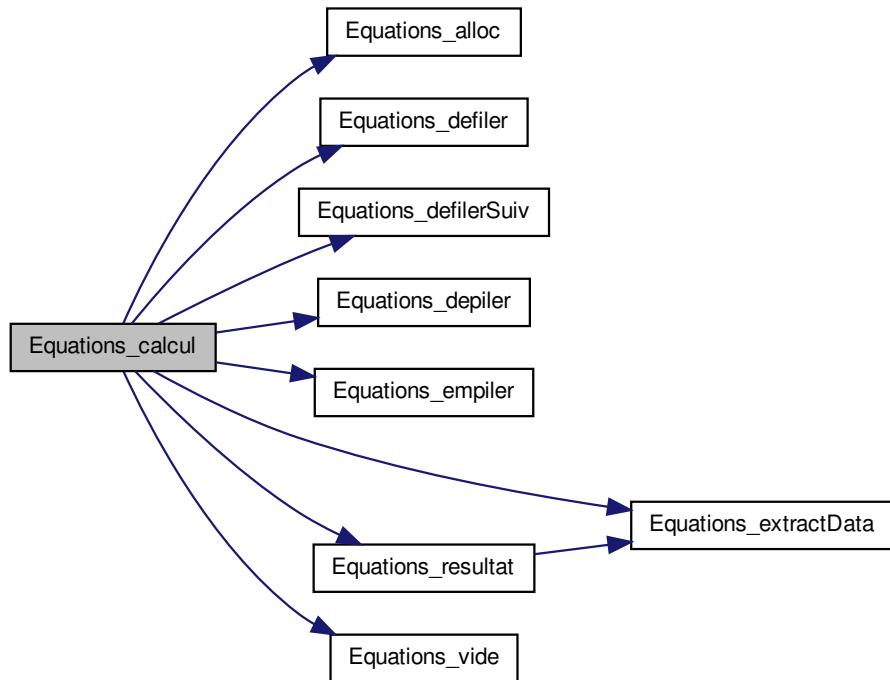
Result of the equation (First part of the energy)

Definition at line 614 of file equations.c.

References addition, constant, division, equal, Equations\_alloc(), Equations\_defiler(), Equations\_defilerSuiv(), Equations\_depiler(), Equations\_empiler(), Equations\_extractData(), Equations\_resultat(), Equations\_vide(), inferior, inferior\_equal, multiplication, soustraction, superior, and superior\_equal.

Referenced by Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

Here is the call graph for this function:



#### 4.12.2.7 pEquations Equations\_defiler ( pEquations liste )

Look for the first element of the struct `Equations`.

##### Author

Amine Ghozlane

##### Parameters

<code>liste</code>	Struct <code>Equations</code>
--------------------	-------------------------------

##### Returns

Pointer on the first element of struct `Equations`

Definition at line 320 of file equations.c.

Referenced by `Equations_calcul()`, and `Equations_emptyOp()`.

**4.12.2.8 pEquations Equations\_defilerSuiv ( pEquations liste )**

Get the next element of the struct [Equations](#).

**Author**

Amine Ghozlane

**Parameters**

<i>liste</i>	Struct <a href="#">Equations</a>
--------------	----------------------------------

**Returns**

Pointer on the next element of struct [Equations](#)

Definition at line 334 of file equations.c.

Referenced by [Equations\\_calcul\(\)](#), and [Equations\\_emptyOp\(\)](#).

**4.12.2.9 void Equations\_define ( pEquations new, char \* operateur )**

Identify the mathematical operator used.

**Author**

Amine Ghozlane

**Parameters**

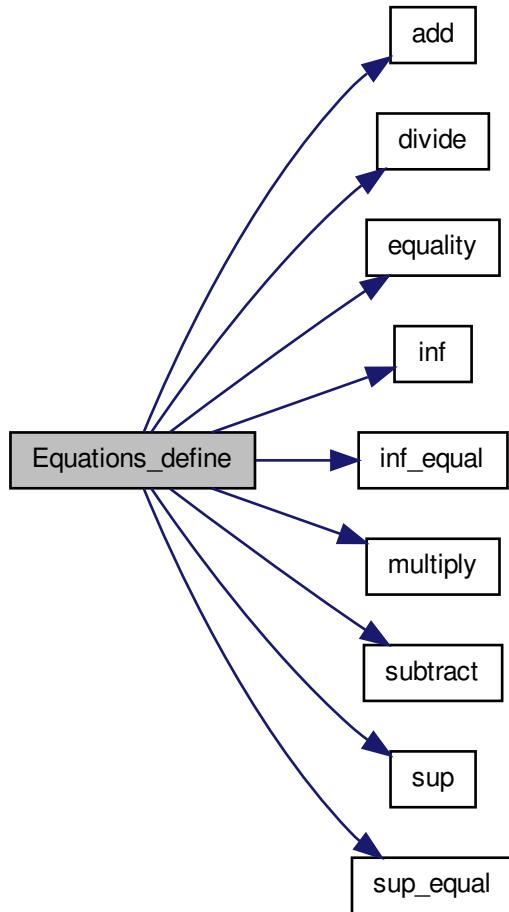
<i>new</i>	Struct <a href="#">Equations</a>
<i>operateur</i>	Read operator

Definition at line 174 of file equations.c.

References [add\(\)](#), [addition](#), [divide\(\)](#), [division](#), [equal](#), [equality\(\)](#), [inf\(\)](#), [inf\\_equal\(\)](#), [inferior](#), [inferior\\_equal](#), [multiplication](#), [multiply\(\)](#), [soustraction](#), [subtraction](#), [subtract\(\)](#), [sup\(\)](#), [sup\\_equal\(\)](#), [superior](#), and [superior\\_equal](#).

Referenced by [Equations\\_pileFormation\(\)](#).

Here is the call graph for this function:



#### 4.12.2.10 pEquations Equations\_depiler( pEquations liste )

Unstack the last element of the struct [Equations](#).

##### Author

Amine Ghozlane

##### Parameters

<i>liste</i>	Struct <a href="#">Equations</a> .
--------------	------------------------------------

**Returns**

Pointer on the "unstack" element of struct [Equations](#).

Definition at line 282 of file equations.c.

Referenced by [Equations\\_addOp\(\)](#), and [Equations\\_calcul\(\)](#).

**4.12.2.11 pEquations Equations.empiler ( pEquations *liste*, pEquations *new* )**

Stack an element to the struct [Equations](#).

**Author**

Amine Ghozlane

**Parameters**

<i>liste</i>	Struct <a href="#">Equations</a>
<i>new</i>	New struct <a href="#">Equations</a> element

**Returns**

Pointer on the last element of struct [Equations](#)

Definition at line 258 of file equations.c.

Referenced by [Equations\\_addOp\(\)](#), [Equations\\_calcul\(\)](#), [Equations\\_emptyOp\(\)](#), and [Equations\\_pileFormation\(\)](#).

**4.12.2.12 void Equations.emptyOp ( pEquations *result*, pEquations *op* )**

Empty the Struct [Equations](#) op for the struct Equation result.

**Author**

Amine Ghozlane

**Parameters**

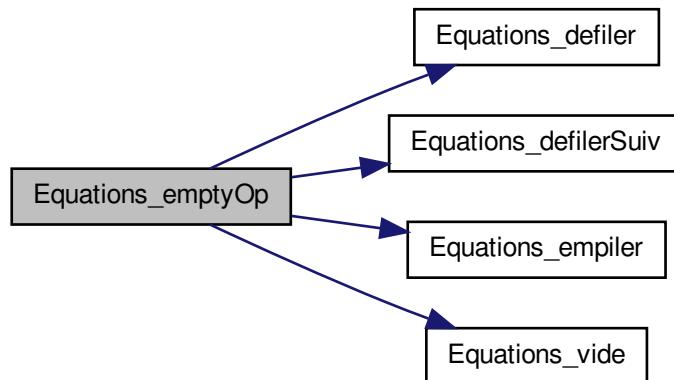
<i>result</i>	Struct <a href="#">Equations</a> used for computation
<i>op</i>	Struct Equation used to store the operator

Definition at line 421 of file equations.c.

References [Equations\\_defiler\(\)](#), [Equations\\_defilerSuiv\(\)](#), [Equations\\_empiler\(\)](#), and [Equations\\_vide\(\)](#).

Referenced by [Equations\\_pileFormation\(\)](#).

Here is the call graph for this function:



**4.12.2.13 double Equations\_extractData ( pEquations *new*, char \*\* *name*, double \* *quantite*, int *taille* )**

Extract information on element of type Struct Equation.

#### Author

Amine Ghozlane

#### Parameters

<i>new</i>	Element of type Struct Equation
<i>name</i>	Name of the interest element
<i>quantite</i>	List of the quantity of the molecules
<i>taille</i>	Number of molecules in the list

#### Returns

Quantity of the molecule of interest

Definition at line 497 of file equations.c.

References constant.

Referenced by Equations\_calcul(), and Equations\_resultat().

---

```
4.12.2.14 double Equations_finalQuantite ( int file_nb_especies, char ** file_species, int *
file_amount, int * file_weight, char ** sim_name, double * sim_quantite, int
sim_taille )
```

Compute the score of the quantity which means the difference between what is expected to what is simulated.

#### Author

Amine Ghozlane

#### Parameters

<i>file_nb_- especes</i>	Number of species in the parameter file
<i>file_species</i>	List of species in the parameter file
<i>file_amount</i>	List of species expected quantity
<i>file_weight</i>	Weight defined for the species
<i>sim_name</i>	List of species simulated (sbml file)
<i>sim_quantite</i>	List of the quantity of the species simulated (sbml file)
<i>sim_taille</i>	Number of species simulated (sbml file)

#### Returns

Result of the difference (Second part of the Energy)

Definition at line 683 of file equations.c.

References Equations\_findSpecies().

Referenced by Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), and Sd\_compute\_simulation().

Here is the call graph for this function:




---

```
4.12.2.15 double Equations_findSpecies ( char * species, char ** name, double * quantite, int
taille )
```

Look for the quantity of a molecule in the list.

**Author**

Amine Ghozlane

**Parameters**

<i>species</i>	Name of a molecule
<i>name</i>	List of molecules
<i>quantite</i>	List of the quantity of the molecules
<i>taille</i>	Number of molecules in the list

**Returns**

Quantity of the molecule of interest

Definition at line 533 of file equations.c.

Referenced by Equations\_finalQuantite(), Min\_score\_print\_mean(), and Mod\_score\_-print\_mean().

**4.12.2.16 pEquations Equations\_pileFormation ( char \*\*\* *equation*, int *nb\_noeud* )**

Build pile.

**Author**

Amine Ghozlane

**Parameters**

<i>equation</i>	Table with the equation in Mathml format
<i>nb_noeud</i>	Number of element inside the equation

**Returns**

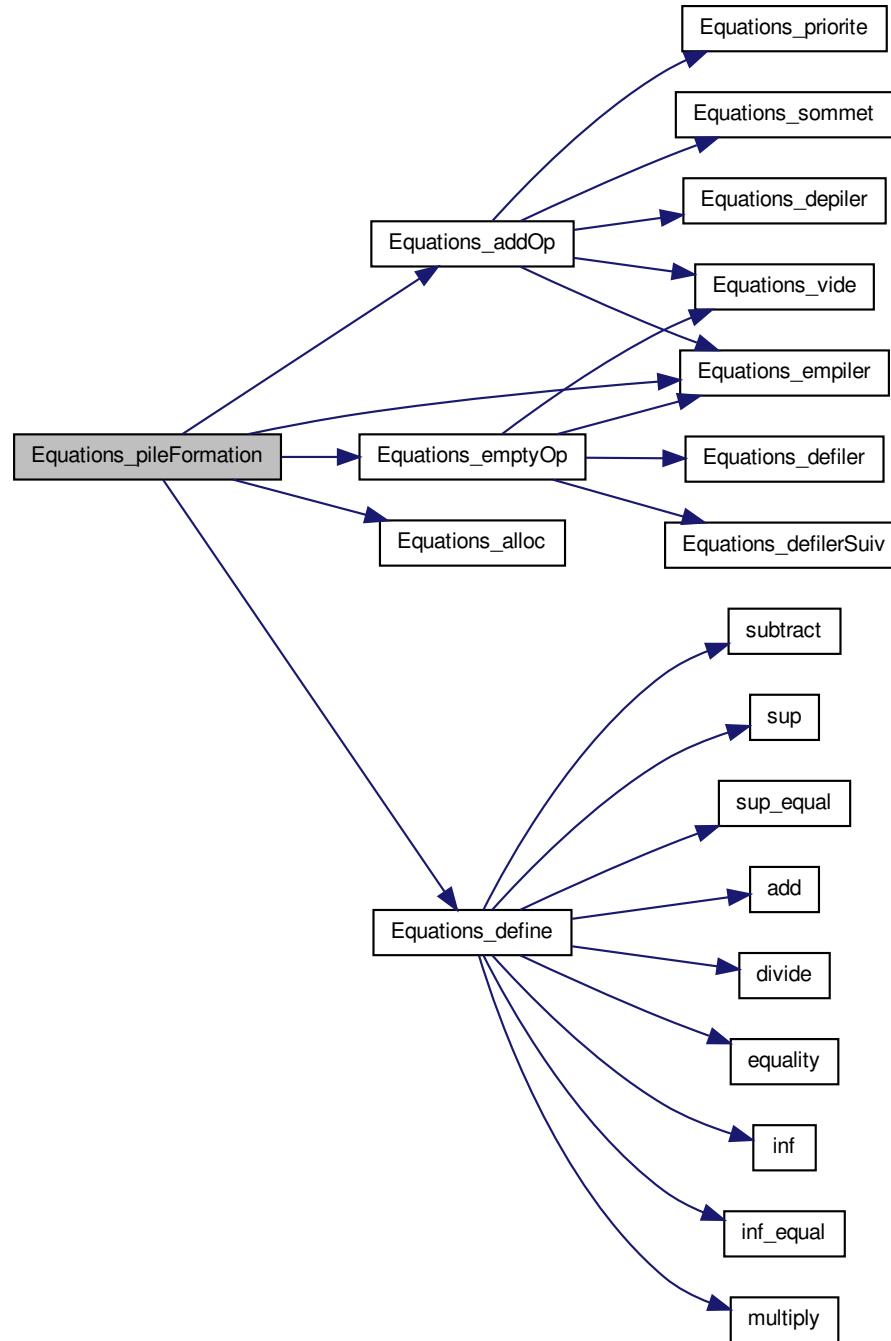
Struct Equation needed for compute the equation

Definition at line 441 of file equations.c.

References constant, Equations\_addOp(), Equations\_alloc(), Equations\_define(), Equations\_-empiler(), Equations\_emptyOp(), and variable.

Referenced by Data\_equationsInit().

Here is the call graph for this function:



**4.12.2.17 void Equations\_print ( pEquations liste )**

Print all information on the element of the struct [Equations](#).

**Author**

Amine Ghozlane

**Parameters**

<i>liste</i>	Struct <a href="#">Equations</a>
--------------	----------------------------------

Definition at line 359 of file equations.c.

References constant, Equdata::data, Equations::info, Equations::type, Equdata::var, and variable.

**4.12.2.18 int Equations\_priorite ( pEquations new )**

Give the priority of the operator. Multiplication and Division have higher priority than addition or subtraction...

**Author**

Amine Ghozlane

**Parameters**

<i>new</i>	One element of the struct <a href="#">Equations</a>
------------	---

**Returns**

Priority of the operator

Definition at line 346 of file equations.c.

References addition, equal, inferior, inferior\_equal, soustraction, superior, and superior\_equal.

Referenced by Equations\_addOp().

**4.12.2.19 double Equations\_resultat ( pEquations result, pEquations result1, pEquations eq, char \*\* name, double \* quantite, int taille )**

Compute the result of two pile.

**Author**

Amine Ghozlane

**Parameters**

<i>result</i>	Struct Equation result
---------------	------------------------

<i>result1</i>	Struct Equation result after the operator
<i>eq</i>	Operator
<i>name</i>	List of molecules
<i>quantite</i>	List of the quantity of the molecules
<i>taille</i>	Number of molecules in the list

**Returns**

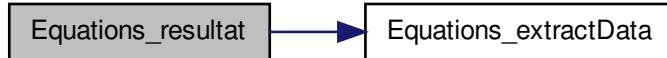
Result of the two pile

Definition at line 561 of file equations.c.

References equal, Equations\_extractData(), inferior, inferior\_equal, superior, superior\_-equal, and Equations::type.

Referenced by Equations\_calcul().

Here is the call graph for this function:

**4.12.2.20 pEquations Equations\_sommet ( pEquations liste )**

Look for the last element of the struct [Equations](#).

**Author**

Amine Ghozlane

**Parameters**

<i>liste</i>	Struct <a href="#">Equations</a>
--------------	----------------------------------

**Returns**

Pointer on the last element of struct [Equations](#)

Definition at line 305 of file equations.c.

Referenced by Equations\_addOp().

**4.12.2.21 int Equations\_vide ( pEquations liste )**

Test if the struct [Equations](#) is empty.

**Author**

Amine Ghozlane

**Parameters**

<i>liste</i>	Struct <a href="#">Equations</a>
--------------	----------------------------------

**Returns**

Number of elements

Definition at line 237 of file equations.c.

Referenced by [Equations\\_addOp\(\)](#), [Equations\\_calcul\(\)](#), and [Equations\\_emptyOp\(\)](#).

**4.12.2.22 double inf ( double arg1, double arg2 )**

Test the inferiority between two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 134 of file equations.c.

Referenced by [Equations\\_define\(\)](#).

**4.12.2.23 double inf\_equal ( double arg1, double arg2 )**

Test the inferiority or equality between two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 147 of file equations.c.

Referenced by Equations\_define().

**4.12.2.24 double multiply ( double arg1, double arg2 )**

Multiply two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 69 of file equations.c.

Referenced by Equations\_define().

**4.12.2.25 double subtract ( double arg1, double arg2 )**

Subtraction two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 56 of file equations.c.

Referenced by Equations\_define().

**4.12.2.26 double sup ( double arg1, double arg2 )**

Test the superiority between two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 108 of file equations.c.

Referenced by Equations\_define().

**4.12.2.27 double sup\_equal ( double *arg1*, double *arg2* )**

Test the superiority or equality between two numbers.

**Author**

Amine Ghozlane

**Parameters**

<i>arg1</i>	Value 1
<i>arg2</i>	Value 2

**Returns**

Result

Definition at line 121 of file equations.c.

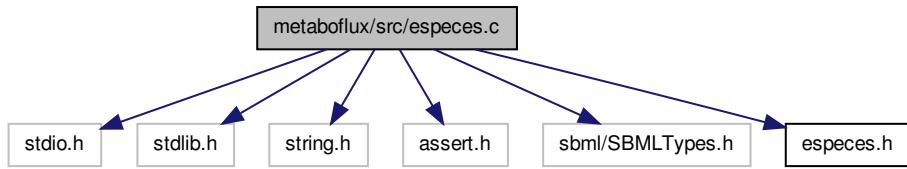
Referenced by Equations\_define().

## 4.13 metaboflux/src/especies.c File Reference

Modelize a molecule.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <sbml/SBMLTypes.h>
#include "especies.h"
```

Include dependency graph for especes.c:



## Functions

- `pEspecies Especies_alloc (int nbEspecies)`  
*Alloc memory and initialize the struct `Especies`.*
- `void Especies_allocReactions (pEspecies molecules, int Espece, Reaction_t *local, double reactRatio)`  
*Alloc memory and initialize the struct `Reaction` for each molecule.*
- `void Especies_free (pEspecies molecules, int nbEspecies)`  
*Free the struct `pEspecies`.*
- `void Especies_freeReactions (Especies molecules)`  
*Free the struct `Reaction`.*
- `void Especies_save (pEspecies molecules, int i, double quant, const char *dye)`  
*Save data on one molecule.*
- `void Especies_scoreSpecies (pEspecies molecules, int nbEspecies, char **name, double *quantite)`  
*Save the quantite of all molecules for scoring.*
- `void Especies_print (pEspecies molecules, int nbEspecies)`  
*Print data on each molecule.*
- `void Especies_print_2 (pEspecies molecules, int nbEspecies)`  
*Print Id and quantity on each molecule.*
- `int Especies_find (pEspecies molecules, const char *especeld, int nbEspecies)`  
*Find a molecule thank to its ID.*
- `void Especies_setQuantite (pEspecies molecules, int Especies, double quant)`  
*Change the quantity of a molecule.*
- `double Especies_getQuantite (pEspecies molecules, int Especies)`  
*Get the quantity of one molecule by its number.*
- `int Especies_getNbreactions (pEspecies molecules, int ref)`  
*Get number of reaction where one molecule is used (like a reactif)*

### 4.13.1 Detailed Description

Modelize a molecule. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflux/>)  
Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

{Amine Ghozlane}

#### Version

2.0

#### Date

27 octobre 2009

Definition in file [especies.c](#).

### 4.13.2 Function Documentation

#### 4.13.2.1 pEspecies Especies.alloc ( int nbEspecies )

Alloc memory and initialize the struct [Especies](#).

#### Author

Amine Ghozlane

#### Parameters

<code>nbEspecies</code>	Number of molecules
-------------------------	---------------------

#### Returns

Allocated struct [Especie](#)

Definition at line 40 of file [especies.c](#).

References [Especies::quantite](#), and [Especies::system](#).

Referenced by [SBML\\_compute\\_simulation\(\)](#), and [SBML\\_compute\\_simulation\\_mean\(\)](#).

---

**4.13.2.2 void Especies\_allocReactions ( pEspecies *molecules*, int *Espece*, Reaction\_t \* *local*, double *reactRatio* )**

Alloc memory and initialize the struct [Reaction](#) for each molecule.

#### Author

Amine Ghozlane

#### Parameters

<i>molecules</i>	List of molecules
<i>Espece</i>	Number of the selected molecule
<i>local</i>	<a href="#">Reaction</a> sbml reference
<i>reactRatio</i>	Ratio of the reaction

Definition at line 64 of file especes.c.

References [Reaction::link](#), [Reaction::suivant](#), and [Especies::system](#).

Referenced by [SBML\\_setReactions\(\)](#).

**4.13.2.3 int Especies\_find ( pEspecies *molecules*, const char \* *especeld*, int *nbEspecies* )**

Find a molecule thank to it ID.

#### Author

Amine Ghozlane

#### Parameters

<i>molecules</i>	List of molecules
<i>especeld</i>	SBML ID of the molecule
<i>nbEspecies</i>	Number of molecules

#### Returns

Number of a selected molecule

Definition at line 214 of file especes.c.

Referenced by [SBML\\_checkQuantite\(\)](#), [SBML\\_reaction\(\)](#), and [SBML\\_setReactions\(\)](#).

**4.13.2.4 void Especies\_free ( pEspecies *molecules*, int *nbEspecies* )**

Free the struct pEspecies.

#### Author

Amine Ghozlane

**Parameters**

<i>molecules</i>	List of molecules
<i>nbEspecies</i>	Number of molecules

Definition at line 94 of file especes.c.

References `Especies_freeReactions()`.

Referenced by `SBML_compute_simulation()`, and `SBML_compute_simulation_mean()`.

Here is the call graph for this function:

**4.13.2.5 void Especies\_freeReactions ( Especies *molecules* )**

Free the struct [Reaction](#).

**Author**

Amine Ghozlane

**Parameters**

<i>molecules</i>	List of molecules
------------------	-------------------

Definition at line 109 of file especes.c.

References `Reaction::suivant`, and `Especies::system`.

Referenced by `Especies_free()`.

**4.13.2.6 int Especies\_getNbReactions ( pEspecies *molecules*, int *ref* )**

Get number of reaction where one molecule is used (like a reactif)

**Author**

Amine Ghozlane

**Parameters**

<i>molecules</i>	List of molecules
<i>ref</i>	Number of a selected molecule

**Returns**

Number of reactions

Definition at line 265 of file especes.c.

References Reaction::suivant, and Especies::system.

Referenced by SBML\_simulate().

**4.13.2.7 double Especies\_getQuantite ( pEspecies *molecules*, int *Especies* )**

Get the quantity of one molecule by its number.

**Author**

Amine Ghozlane

**Parameters**

<i>molecules</i>	List of molecules
<i>Especies</i>	Number of a selected molecule

**Returns**

Quantity of a selected molecule

Definition at line 252 of file especes.c.

References Especies::quantite.

Referenced by SBML\_checkQuantite(), SBML\_reaction(), and SBML\_simulate().

**4.13.2.8 void Especies\_print ( pEspecies *molecules*, int *nbEspecies* )**

Print data on each molecule.

**Author**

Amine Ghozlane

**Parameters**

<i>molecules</i>	List of molecules
<i>nbEspecies</i>	Number of molecules

Definition at line 170 of file especes.c.

References Reaction::link, Reaction::suivant, and Especies::system.

**4.13.2.9 void Especies\_print\_2 ( pEspecies *molecules*, int *nbEspecies* )**

Print Id and quantity on each molecule.

**Author**

Amine Ghozlane

**Parameters**

<i>molecules</i>	List of molecules
<i>nbEspecies</i>	Number of molecules

Definition at line 196 of file especes.c.

**4.13.2.10 void Especies.save ( pEspecies *molecules*, int *i*, double *quant*, const char \* *dye* )**

Save data on one molecule.

**Author**

Amine Ghozlane

**Parameters**

<i>molecules</i>	List of molecules
<i>i</i>	Number of the selected molecule
<i>quant</i>	Quantity of the molecule
<i>dye</i>	ID of the molecule

Definition at line 134 of file especes.c.

References Especies::id, and Especies::quantite.

Referenced by SBML\_initEspeceAmounts().

**4.13.2.11 void Especies.scoreSpecies ( pEspecies *molecules*, int *nbEspecies*, char \*\* *name*, double \* *quantite* )**

Save the quantite of all molecules for scoring.

**Author**

Amine Ghozlane

**Parameters**

<i>molecules</i>	List of molecules
<i>nbEspecies</i>	Number of molecules
<i>name</i>	ID of the selected molecule
<i>quantite</i>	Table of molecule Quantity

Definition at line 149 of file especes.c.

References Especies::quantite.

Referenced by SBML\_score().

#### 4.13.2.12 void Especies\_setQuantite ( pEspecies *molecules*, int *Especies*, double *quant* )

Change the quantity of a molecule.

##### Author

Amine Ghozlane

##### Parameters

<i>molecules</i>	List of molecules
<i>Especies</i>	Number of a selected molecule
<i>quant</i>	Quantity of the molecule

Definition at line 239 of file especes.c.

References *Especies*::quantite.

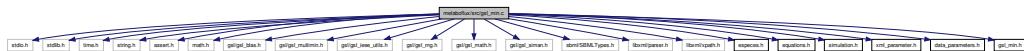
Referenced by SBML\_reaction().

## 4.14 metaboflux/src/gsl\_min.c File Reference

Compute minimization of scenarii.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <assert.h>
#include <math.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_multimin.h>
#include <gsl/gsl_ieee_utils.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_siman.h>
#include <sbml/SBMLTypes.h>
#include <libxml/parser.h>
#include <libxml>xpath.h>
#include "especies.h"
#include "equations.h"
#include "simulation.h"
```

```
#include "xml_parameter.h"  
#include "data_parameters.h"  
#include "gsl_min.h"
```



## Functions

- int [Min\\_copieTab2](#) (double \*x, int current, int debut, int fin)  
*Copy table of reaction parameters.*
  - int [Min\\_copieTab3](#) (gsl\_multimin\_fminimizer \*s, int current, int debut, int fin)  
*Copy short table of parameter into the larger table.*
  - void [Min\\_verifyValue](#) (gsl\_vector \*v, double \*x, int debut, int max)  
*Checking reaction ratio parameters.*
  - double [Min\\_my\\_f](#) (const gsl\_vector \*v, void \*params)  
*Enter in the program for standard deviation.*
  - void [Min\\_getTampon](#) (double \*energie\_temp, double energie)  
*Enter in the program for standard deviation.*
  - void [Min\\_score\\_print\\_mean](#) (double \*result\_tab)  
*Save the minimization result.*
  - void [Min\\_compute\\_minimization](#) (pListParameters allone, double \*fluxRatio, double \*result\_tab, char \*\*files\_path, int number\_arg, int debug)  
*Compute the minimization.*

#### 4.14.1 Detailed Description

Compute minimization of scenarii. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflux>).  
Copyright (C) 2010 Amine Ghoulane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

## Author

{Amine Ghozlane}

**Version**

2.0

**Date**

9 novembre 2009

Definition in file [gsl\\_min.c](#).**4.14.2 Function Documentation****4.14.2.1 void Min\_compute\_minimization ( pListParameters *allone*, double \* *fluxRatio*, double \* *result\_tab*, char \*\* *files\_path*, int *number\_arg*, int *debug* )**

Compute the minimization.

**Author**

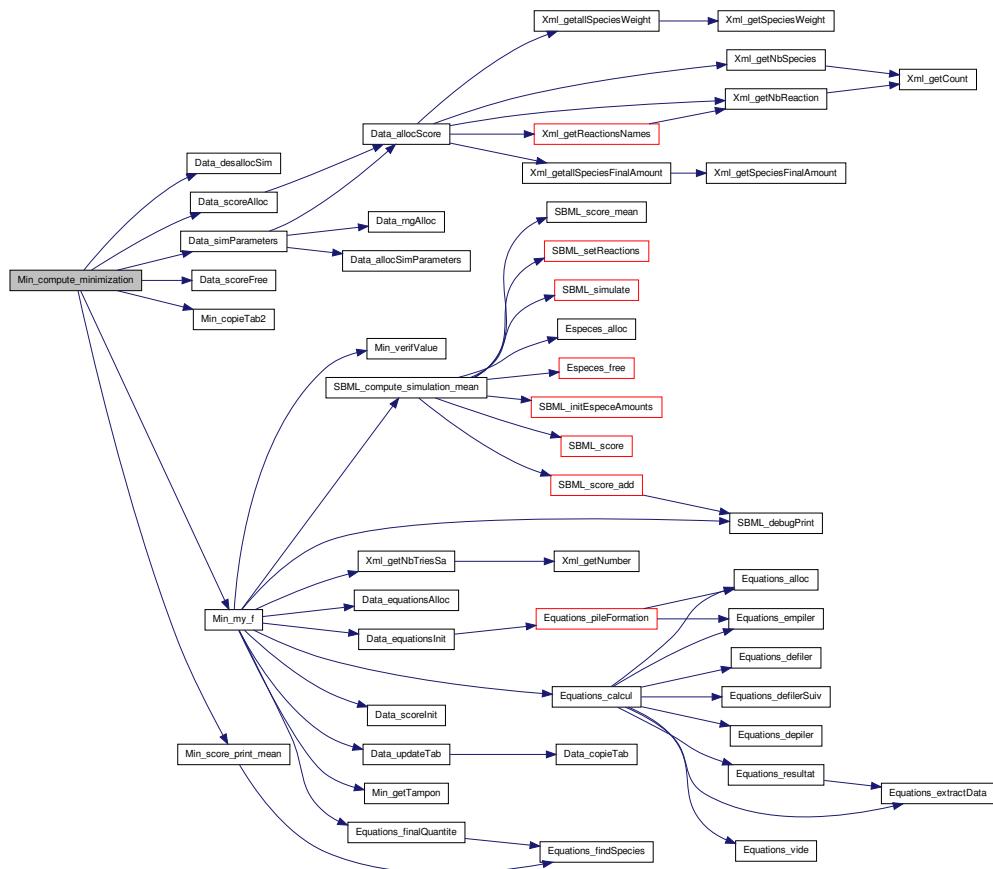
Amine Ghozlane

**Parameters**

<i>allone</i>	struct <a href="#">ListParameters</a>
<i>fluxRatio</i>	Ratio parameters
<i>result_tab</i>	Result table
<i>files_path</i>	List of paths
<i>number_arg</i>	Number of simulation
<i>debug</i>	Debug flag

Definition at line 274 of file [gsl\\_min.c](#).References [Data\\_desallocSim\(\)](#), [Data\\_scoreAlloc\(\)](#), [Data\\_scoreFree\(\)](#), [Data\\_simParameters\(\)](#), [ListParameters::interest\\_parameters](#), [Min\\_copieTab2\(\)](#), [Min\\_my\\_f\(\)](#), [Min\\_score\\_print\\_mean\(\)](#), [ListParameters::nb\\_couples](#), [ListParameters::nb\\_parameters](#), [ListParameters::parameters](#), and [SimParameters::y](#).Referenced by [Mpi\\_slave\(\)](#).

Here is the call graph for this function:



#### 4.14.2.2 int Min\_copieTab2 ( double \* x, int current, int debut, int fin )

Copy table of reaction parameters.

## Author

Amine Ghozlane

## Parameters

<i>x</i>	Short table of reaction parameters
<i>current</i>	Line
<i>debut</i>	Beginning
<i>fin</i>	End

**Returns**

Number of copied element

Definition at line 68 of file `gsl_min.c`.

References `SimParameters::y`.

Referenced by `Min_compute_minimization()`.

**4.14.2.3 int Min\_copieTab3 ( `gsl_multimin_fminimizer * s`, `int current`, `int debut`, `int fin` )**

Copy short table of parameter into the larger table.

**Author**

Amine Ghozlane

**Parameters**

<code>s</code>	Minimizer parameter
<code>current</code>	Line
<code>debut</code>	Beginning
<code>fin</code>	End

**Returns**

Number of copied element

Definition at line 90 of file `gsl_min.c`.

References `SimParameters::y`.

**4.14.2.4 void Min\_getTampon ( `double * energie_temp`, `double energie` )**

Enter in the program for standard deviation.

**Author**

Amine Ghozlane

**Parameters**

<code>energie_-temp</code>	Current energy
<code>energie</code>	Best energy

Definition at line 208 of file `gsl_min.c`.

References `ListParameters::nb_parameters`, `SimParameters::out`, `Score::quantite`, `Score::taille`, and `SimParameters::y`.

Referenced by `Min_my_f()`.

**4.14.2.5 double Min\_my\_f( const gsl\_vector \* v, void \* params )**

Enter in the program for standard deviation.

**Author**

Amine Ghozlane

**Parameters**

v	Vector of reaction parameters
params	Unused parameter define by GSL

**Returns**

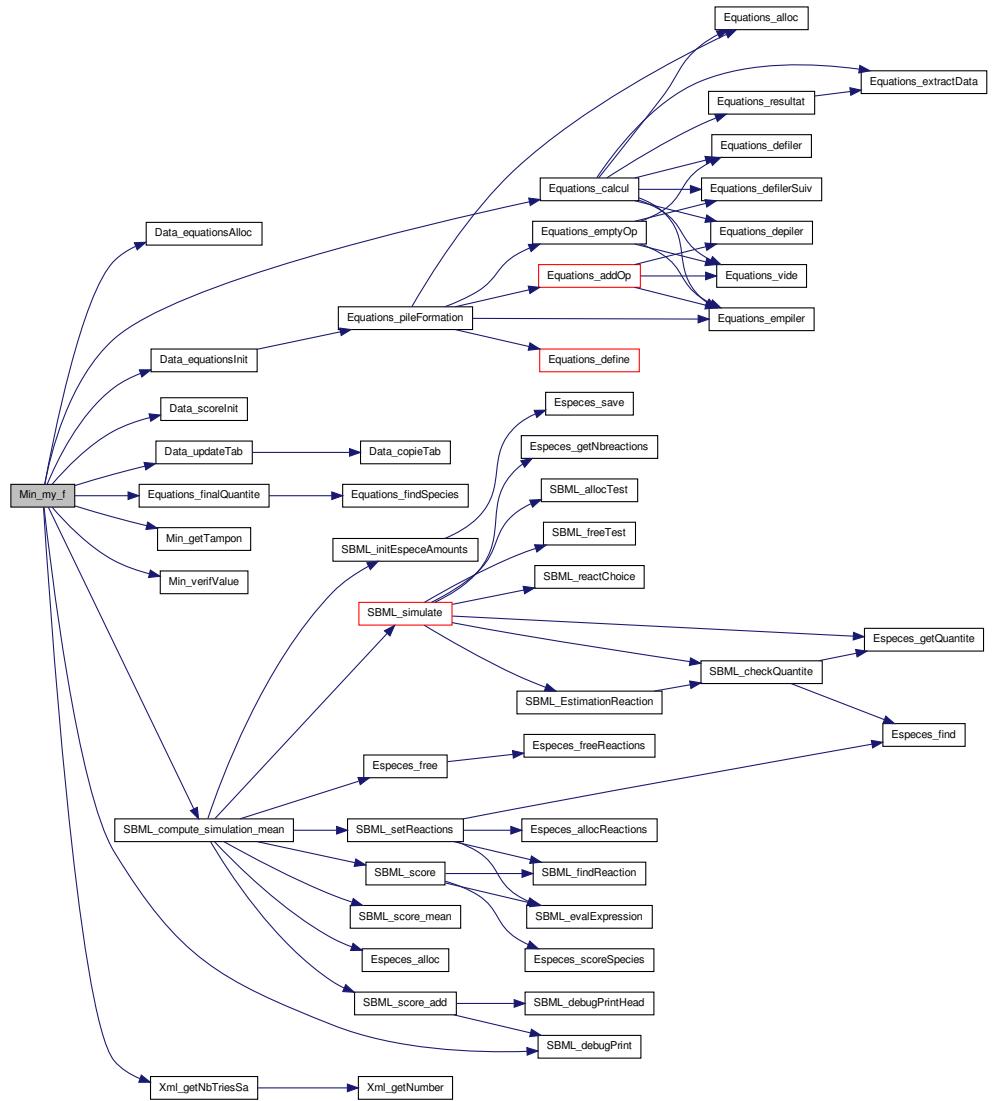
Energy value

Definition at line 149 of file gsl\_min.c.

References ListParameters::banned, ListParameters::conf, Data\_equationsAlloc(), Data\_equationsInit(), Data\_scoreInit(), Data\_updateTab(), SimParameters::debugFile, Equations\_calcul(), Equations\_finalQuantite(), Min\_getTampon(), Min\_verifValue(), ListParameters::model, Score::name, ListParameters::nb\_banned, ListParameters::nb\_couples, ListParameters::nb\_equations, ListParameters::nb\_parameters, Score::nb\_species, SimParameters::out, ListParameters::parameters, SimParameters::pile, Score::quantite, SimParameters::r, SBML\_compute\_simulation\_mean(), SBML\_debugPrint(), Score::species, Score::species\_amount, Score::species\_weight, Score::taille, Score::tailleSpecies, Xml\_getNbTriesSa(), and SimParameters::y.

Referenced by Min\_compute\_minimization().

Here is the call graph for this function:



#### 4.14.2.6 void Min\_score\_print\_mean ( double \* result\_tab )

Save the minimization result.

## Author

Amine Ghozlane

**Parameters**

<i>result_tab</i>	Result table
-------------------	--------------

Definition at line 230 of file gsl\_min.c.

References Equations\_findSpecies(), Score::name, ListParameters::nb\_parameters, Score::nb\_species, SimParameters::out, Score::quantite, Score::species, Score::species\_amount, Score::taille, Score::tailleSpecies, and SimParameters::y.

Referenced by Min\_compute\_minimization().

Here is the call graph for this function:



#### 4.14.2.7 void Min\_verifValue ( *gsl\_vector* \* *v*, double \* *x*, int *debut*, int *max* )

Checking reaction ratio parameters.

**Author**

Amine Ghozlane

**Parameters**

<i>v</i>	Vector of reaction parameters
<i>x</i>	Short table of reaction parameters
<i>debut</i>	Beginning
<i>max</i>	End

Definition at line 114 of file gsl\_min.c.

Referenced by Min\_my\_f().

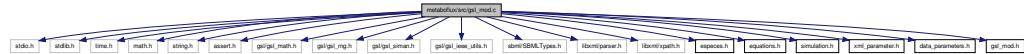
## 4.15 metaboflux/src/gsl\_mod.c File Reference

Compute the modeling of scenarii.

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

```
#include <time.h>
#include <math.h>
#include <string.h>
#include <assert.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_siman.h>
#include <gsl/gsl_ieee_utils.h>
#include <sbml/SBMLTypes.h>
#include <libxml/parser.h>
#include <libxml>xpath.h>
#include "especies.h"
#include "equations.h"
#include "simulation.h"
#include "xml_parameter.h"
#include "data_parameters.h"
#include "gsl_mod.h"
```

Include dependency graph for gsl\_mod.c:



## Functions

- void **Mod\_score\_print\_mean** (pListParameters a, pSimParameters simu, double \*result\_tab, double energie, int number\_group, int group)
 

*Print the mean score.*
- void **Mod\_compute\_modeling** (pListParameters a, double \*fluxRatio, double \*result\_tab, char \*\*files\_path, int number\_group, int group, int debug)
 

*Compute the modeling.*

### 4.15.1 Detailed Description

Compute the modeling of scenarii. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/>)  
 Copyright (C) 2010 Amine Ghazlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation,

either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

{Amine Ghozlane}

#### Version

2.0

#### Date

9 novembre 2009

Definition in file [gsl\\_mod.c](#).

### 4.15.2 Function Documentation

**4.15.2.1 void Mod\_compute\_modeling ( pListParameters *a*, double \* *fluxRatio*, double \* *result\_tab*, char \*\* *files\_path*, int *number\_group*, int *group*, int *debug* )**

Compute the modeling.

#### Author

Amine Ghozlane

#### Parameters

<i>a</i>	struct <a href="#">ListParameters</a>
<i>fluxRatio</i>	Ratio parameters
<i>result_tab</i>	Result table
<i>files_path</i>	List of paths
<i>number_group</i>	Number of the group
<i>group</i>	Group flag
<i>debug</i>	Debug flag

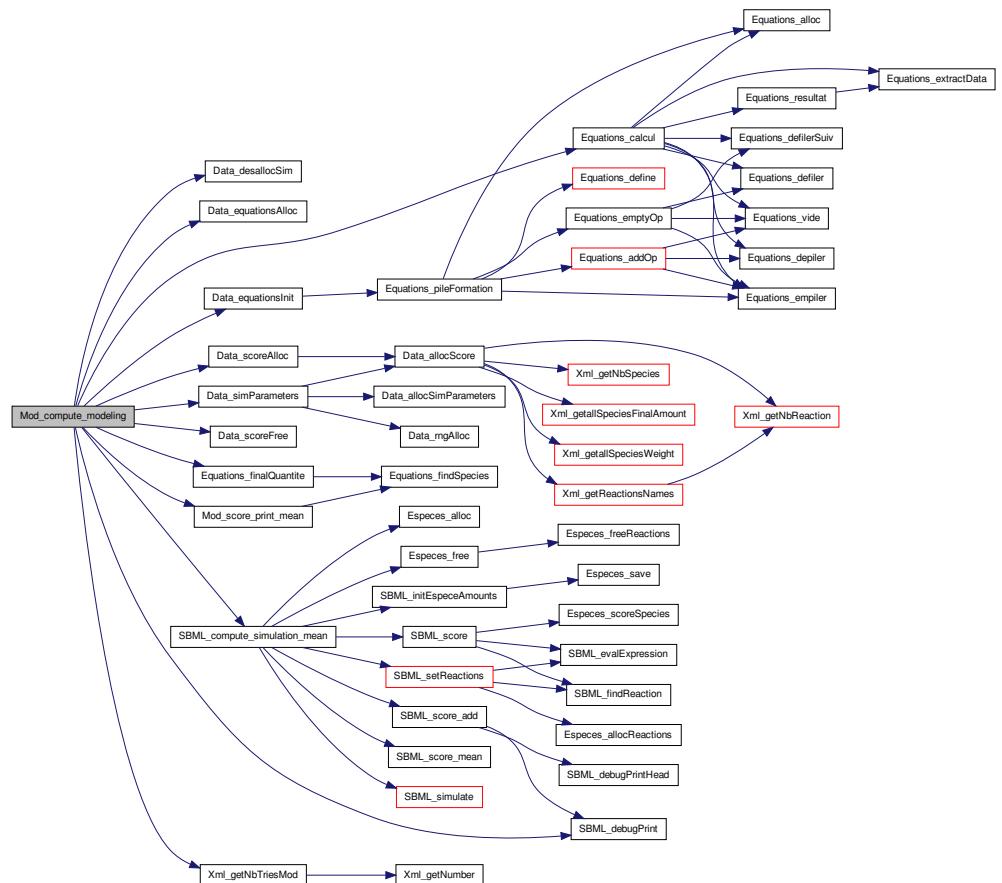
Definition at line 109 of file [gsl\\_mod.c](#).

References [ListParameters::banned](#), [ListParameters::conf](#), [Data\\_desallocSim\(\)](#), [Data\\_equationsAlloc\(\)](#), [Data\\_equationsInit\(\)](#), [Data\\_scoreAlloc\(\)](#), [Data\\_scoreFree\(\)](#), [Data\\_simParameters\(\)](#), [SimParameters::debugFile](#), [Equations\\_calcul\(\)](#), [Equations\\_finalQuantite\(\)](#), [Mod\\_score\\_print\\_mean\(\)](#), [ListParameters::model](#), [Score::name](#), [ListParameters::nb\\_banned](#), [ListParameters::nb\\_equations](#), [ListParameters::nb\\_parameters](#), [Score::nb\\_species](#), [SimParameters::out](#), [Sim-](#)

Parameters::pile, Score::quantite, SimParameters::r, SBML\_compute\_simulation\_mean(), SBML\_debugPrint(), Score::species, Score::species\_amount, Score::species\_weight, Score::taille, Score::tailleSpecies, Xml\_getNbTriesMod(), and SimParameters::y.

Referenced by Mpi\_slave().

Here is the call graph for this function:



**4.15.2.2 void Mod.score\_print\_mean ( pListParameters a, pSimParameters simu, double \* result\_tab, double energie, int number\_group, int group )**

Print the mean score.

#### Author

Amine Ghozlane

**Parameters**

<i>a</i>	Global parameters
<i>simu</i>	Simulation parameters
<i>result_tab</i>	Result table
<i>energie</i>	Energy of the simulation
<i>number_group</i>	Number of simulation
<i>group</i>	Number of the group (if a group is simulated)

Definition at line 58 of file gsl\_mod.c.

References Equations\_findSpecies(), Score::name, ListParameters::nb\_parameters, Score::nb\_species, SimParameters::out, Score::quantite, Score::species, Score::species\_amount, Score::taille, Score::tailleSpecies, and SimParameters::y.

Referenced by Mod\_compute\_modeling().

Here is the call graph for this function:



## 4.16 metaboflux/src/gsl\_recuit.c File Reference

Compute the simulated annealing.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <time.h>
#include <math.h>
#include <sys/types.h>
#include <unistd.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_multimin.h>
#include <gsl/gsl_math.h>

```

```
#include <gsl/gsl_rng.h>
#include <gsl/gsl_siman.h>
#include <gsl/gsl_ieee_utils.h>
#include <sbml/SBMLTypes.h>
#include <libxml/parser.h>
#include <libxml>xpath.h>
#include "xml_parameter.h"
#include "especies.h"
#include "equations.h"
#include "simulation.h"
#include "data_parameters.h"
#include "gsl_recuit.h"

Include dependency graph for gsl_recuit.c:
```

include dependency graph for `gsl_result.c`.



# Functions

- `double Recuit_energyFunction (void *xp)`  
*Simulate the petri net and compute the energy.*
  - `double Recuit_metricDistance (void *xp, void *yp)`  
*Compute the distance between the two table.*
  - `void Recuit_takeStep (const gsl_rng *r, void *xp, double step_size)`  
*Take a step through the solution space TSP.*
  - `void Recuit_printPosition (void *xp)`  
*Print the table of reaction parameters.*
  - `FILE * Recuit_redirectionFlux (xmlConfig_t *conf, FILE *f, char **files_path, int number)`  
*Redirect stdout flux.*
  - `void Recuit_verifParameters (double *x_initial, const gsl_rng *r, int debut, int max)`  
*Choice of parameters and verification of their value.*
  - `void Recuit_verifParameters_2 (double *new_x, double *old_x, const gsl_rng *r, int debut, int max, double step_size)`  
*Choice of parameters and verification of their value.*
  - `void Recuit_defParametre (double *x_initial, const gsl_rng *r)`  
*Definition of the initial parameters of the simulation.*
  - `void Recuit_printParametre (double *x_initial)`

*Print initial parameters.*

- void `Recuit_compute_recuit` (char \*\*`files_path`, int `debug`, int `number`, `pListParameters` `allone`, `gsl_siman_params_t` `params`)

*Compute the simulated annealing.*

#### 4.16.1 Detailed Description

Compute the simulated annealing. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflu>). Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

##### Author

{Amine Ghozlane}

##### Version

2.0

##### Date

9 novembre 2009

Definition in file [gsl\\_recuit.c](#).

#### 4.16.2 Function Documentation

##### 4.16.2.1 void Recuit\_compute\_recuit ( char \*\* files\_path, int debug, int number, pListParameters allone, gsl\_siman\_params\_t params )

Compute the simulated annealing.

##### Author

Amine Ghozlane

##### Parameters

<code>files_path</code>	List of paths
<code>debug</code>	Debug flag
<code>number</code>	Number of the simulation
<code>allone</code>	Global parameters : struct <a href="#">ListParameters</a>
<code>params</code>	Gst parameters

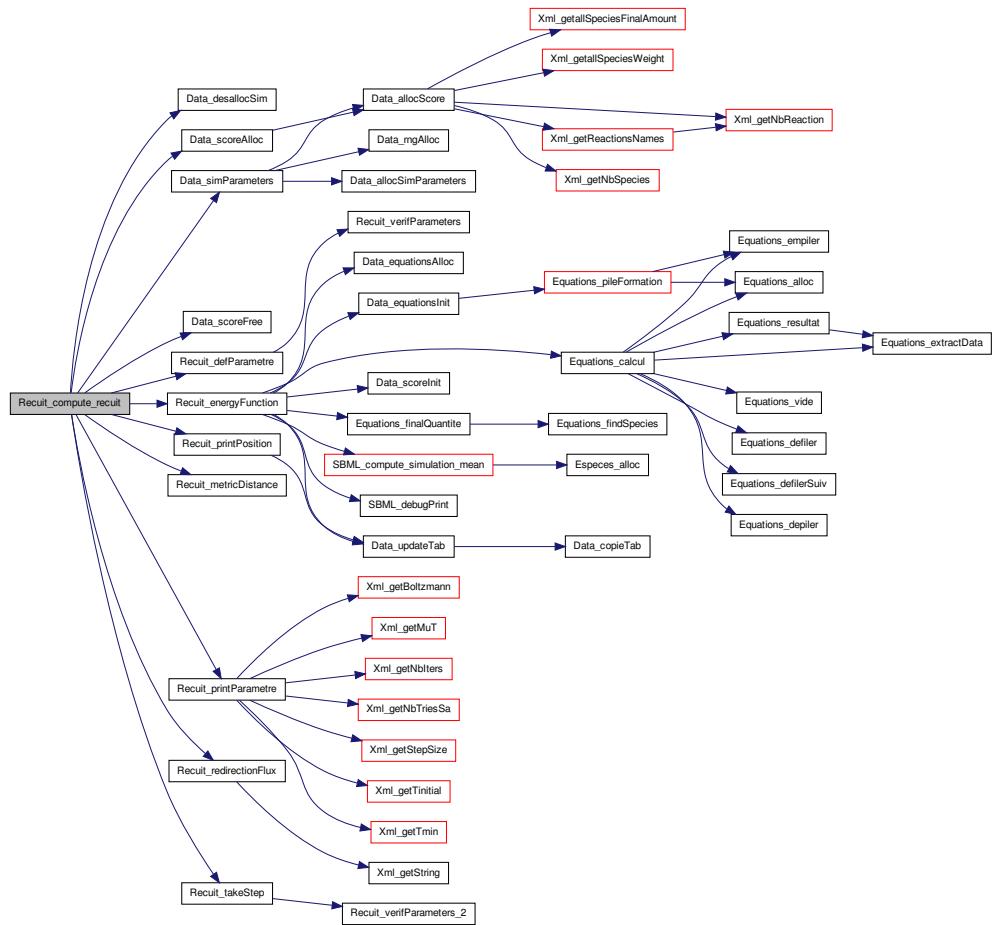
Generated on Wed Apr 27 2011 16:40:50 for Metaboflux by Doxygen

Definition at line 304 of file `gsl_recuit.c`.

References `ListParameters::conf`, `Data_desallocSim()`, `Data_scoreAlloc()`, `Data_scoreFree()`, `Data_simParameters()`, `ListParameters::interest_parameters`, `SimParameters::r`, `Recuit_defParametre()`, `Recuit_energyFunction()`, `Recuit_metricDistance()`, `Recuit_printParametre()`, `Recuit_printPosition()`, `Recuit_redirectionFlux()`, and `Recuit_takeStep()`.

Referenced by `Mpi_slave()`.

Here is the call graph for this function:



#### 4.16.2.2 void Recuit\_defParametre ( double \* *x\_initial*, const gsl\_rng \* *r* )

Definition of the initial parameters of the simulation.

**Author**

Amine Ghozlane

**Parameters**

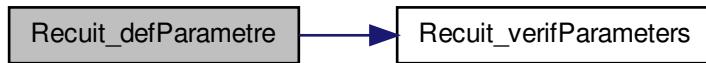
<i>x_initial</i>	table of reaction parameters
<i>r</i>	Random number generator

Definition at line 261 of file gsl\_recuit.c.

References ListParameters::nb\_couples, ListParameters::parameters, and Recuit\_verifParameters().

Referenced by Recuit\_compute\_recuit().

Here is the call graph for this function:



### 4.16.2.3 double Recuit\_energyFunction ( void \* xp )

Simulate the petri net and compute the energy.

double [Recuit\\_energyFunction\(void \\*xp\)](#)

**Author**

Amine Ghozlane

**Parameters**

<i>xp</i>	Short table of reaction parameters
-----------	------------------------------------

**Returns**

Energy value

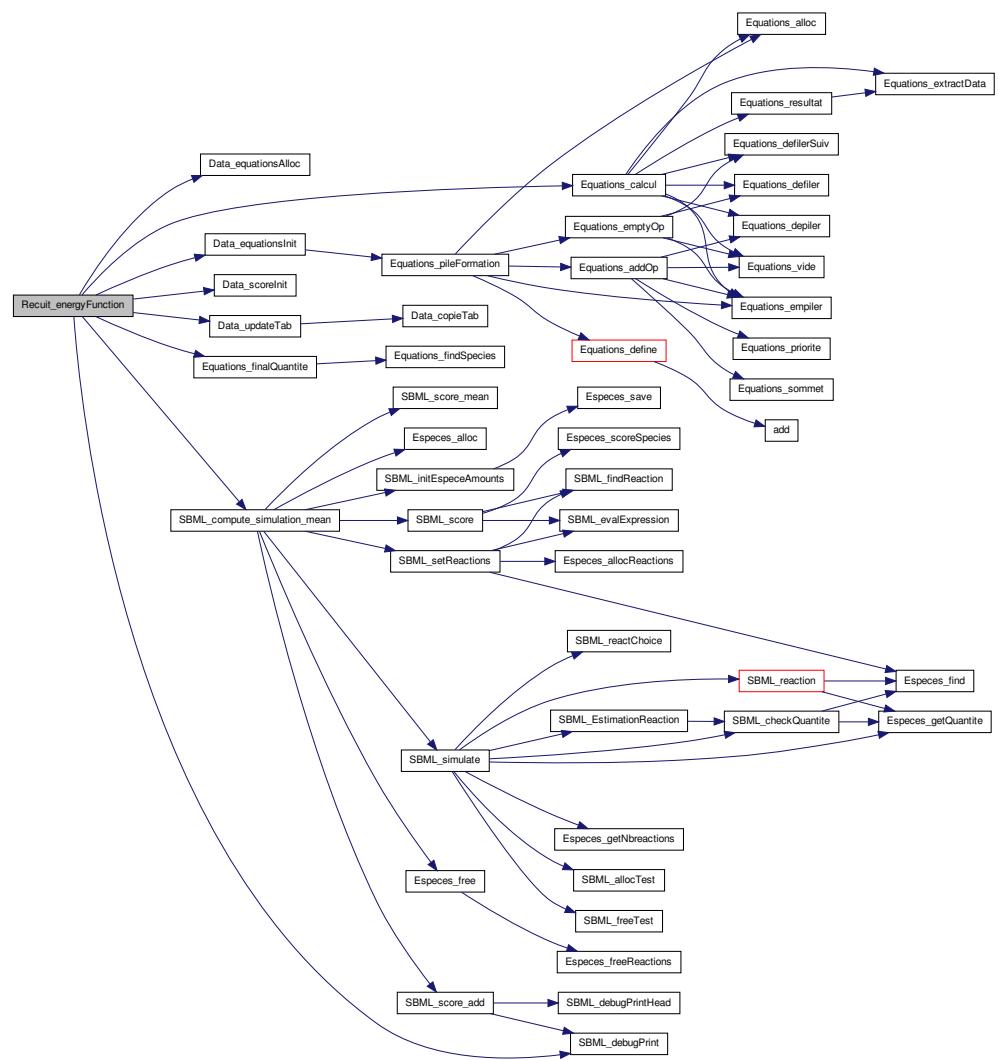
Definition at line 65 of file gsl\_recuit.c.

References ListParameters::banned, Data\_equationsAlloc(), Data\_equationsInit(), Data\_scoreInit(), Data\_updateTab(), SimParameters::debugFile, Equations\_calcul(), Equations\_finalQuantite(), ListParameters::model, Score::name, ListParameters::nb\_banned, ListParameters::nb\_equations, Score::nb\_species, ListParameters::nb\_triesSa, SimParameters::out, SimParameters::pile, Score::quantite, SimParameters::r, SBML\_compute\_simulation\_mean(),

`SBML_debugPrint()`, `Score::species`, `Score::species_amount`, `Score::species_weight`, `Score::taille`, `Score::tailleSpecies`, and `SimParameters::y`.

Referenced by Recuit\_compute\_recuit().

Here is the call graph for this function:



#### 4.16.2.4 double Recuit\_metricDistance ( void \* xp, void \* yp )

Compute the distance between the two table.

**Author**

Amine Ghozlane

**Parameters**

<i>xp</i>	Past short table of reaction parameters
<i>yp</i>	New short table of reaction parameters

**Returns**

Distance between the two table

Definition at line 115 of file gsl\_recuit.c.

References ListParameters::interest\_parameters.

Referenced by Recuit\_compute\_recuit().

**4.16.2.5 void Recuit\_printParametre ( double \* *x\_initial* )**

Print initial parameters.

**Author**

Amine Ghozlane

**Parameters**

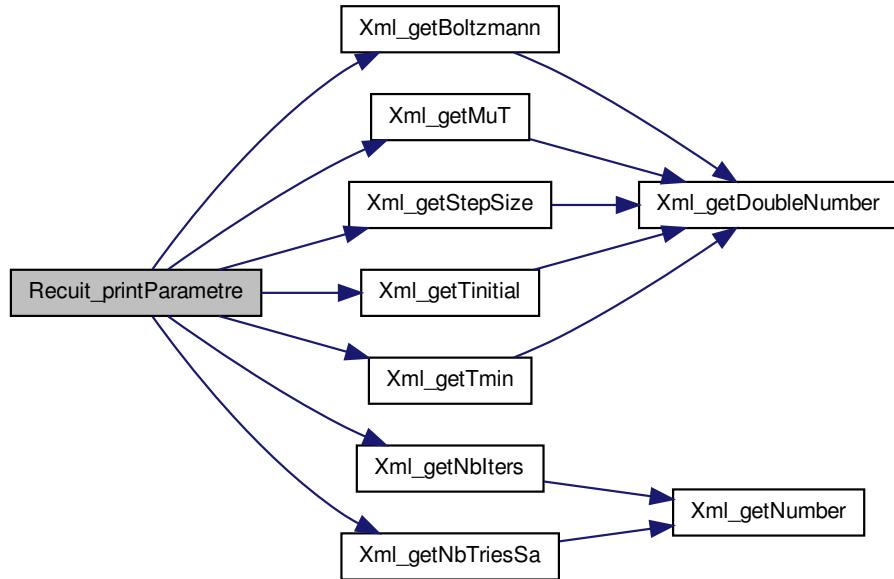
<i>x_initial</i>	table of reaction parameters
------------------	------------------------------

Definition at line 279 of file gsl\_recuit.c.

References ListParameters::conf, ListParameters::interest\_parameters, Xml\_getBoltzmann(), Xml\_getMuT(), Xml\_getNbIter(), Xml\_getNbTriesSa(), Xml\_getStepSize(), Xml\_getTinitial(), and Xml\_getTmin().

Referenced by Recuit\_compute\_recuit().

Here is the call graph for this function:



#### 4.16.2.6 void Recuit\_printPosition ( void \* xp )

Print the table of reaction parameters.

##### Author

Amine Ghozlane

##### Parameters

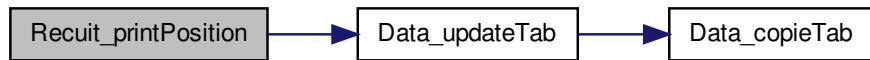
<i>xp</i>	Short table of reaction parameters
-----------	------------------------------------

Definition at line 158 of file gsl\_recuit.c.

References Data\_updateTab(), ListParameters::nb\_parameters, and SimParameters::y.

Referenced by Recuit\_compute\_recuit().

Here is the call graph for this function:



#### 4.16.2.7 FILE \* Recuit\_redirectionFlux ( xmlConfig\_t \* conf, FILE \* f, char \*\* files\_path, int number )

Redirect stdout flux.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>f</i>	File name
<i>files_path</i>	List of paths
<i>number</i>	Index of out repertory

##### Returns

Out flux

Definition at line 184 of file [gsl\\_recuit.c](#).

References [Xml\\_getString\(\)](#).

Referenced by [Recuit\\_compute\\_recuit\(\)](#).

Here is the call graph for this function:



#### 4.16.2.8 void Recuit\_takeStep ( const gsl\_rng \* r, void \* xp, double step\_size )

Take a step through the solution space TSP.

##### Author

Amine Ghozlane

##### Parameters

<i>r</i>	Random number generator
<i>xp</i>	Short table of reaction parameters
<i>step_size</i>	Size of step

Definition at line 136 of file gsl\_recuit.c.

References ListParameters::interest\_parameters, ListParameters::nb\_couples, ListParameters::parameters, and Recuit\_verifParameters\_2().

Referenced by Recuit\_compute\_recuit().

Here is the call graph for this function:



#### 4.16.2.9 void Recuit\_verifParameters ( double \* x\_initial, const gsl\_rng \* r, int debut, int max )

Choice of parameters and verification of their value.

##### Author

Amine Ghozlane

##### Parameters

<i>x_initial</i>	table of reaction parameters
<i>r</i>	Random number generator
<i>debut</i>	Beginning
<i>max</i>	End

Definition at line 207 of file gsl\_recuit.c.

Referenced by Recuit\_defParametre().

---

4.16.2.10 `void Recuit_verifParameters_2 ( double * new_x, double * old_x, const gsl_rng * r, int debut, int max, double step_size )`

Choice of parameters and verification of their value.

#### Author

Amine Ghozlane

#### Parameters

<i>new_x</i>	table of reaction parameters
<i>old_x</i>	table of reaction parameters
<i>r</i>	Random number generator
<i>debut</i>	Beginning
<i>max</i>	End
<i>step_size</i>	Size of the step

Definition at line 235 of file `gsl_recuit.c`.

Referenced by `Recuit_takeStep()`.

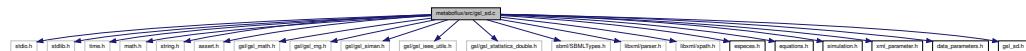
## 4.17 metaboflux/src/gsl\_sd.c File Reference

Compute the standard deviation analysis of the simulations.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <string.h>
#include <assert.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_siman.h>
#include <gsl/gsl_ieee_utils.h>
#include <gsl/gsl_statistics_double.h>
#include <sbml/SBMLTypes.h>
#include <libxml/parser.h>
#include <libxml>xpath.h>
#include "especies.h"
#include "equations.h"
```

```
#include "simulation.h"
#include "xml_parameter.h"
#include "data_parameters.h"
#include "gsl_sd.h"

Include dependency graph for gsl_sd.c:
```



## Functions

- double [Sd\\_compute\\_simulation](#) ([pListParameters](#) all, [pSimParameters](#) sim)  
*Enter in the program for standard deviation.*
- void [Sd\\_compute\\_standard\\_deviation](#) ([pListParameters](#) a, double \*fluxRatio, double \*result\_tab, char \*\*files\_path, int number\_arg, int debug)  
*Compute the standard deviation.*

### 4.17.1 Detailed Description

Compute the standard deviation analysis of the simulations. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflux/>) Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

{Amine Ghozlane}

#### Version

2.0

#### Date

9 novembre 2009

Definition in file [gsl\\_sd.c](#).

## 4.17.2 Function Documentation

### 4.17.2.1 double Sd\_compute\_simulation ( pListParameters *all*, pSimParameters *sim* )

Enter in the program for standard deviation.

#### Author

Amine Ghozlane

#### Parameters

<i>all</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>sim</i>	Simulation parameters : struct <a href="#">SimParameters</a>

#### Returns

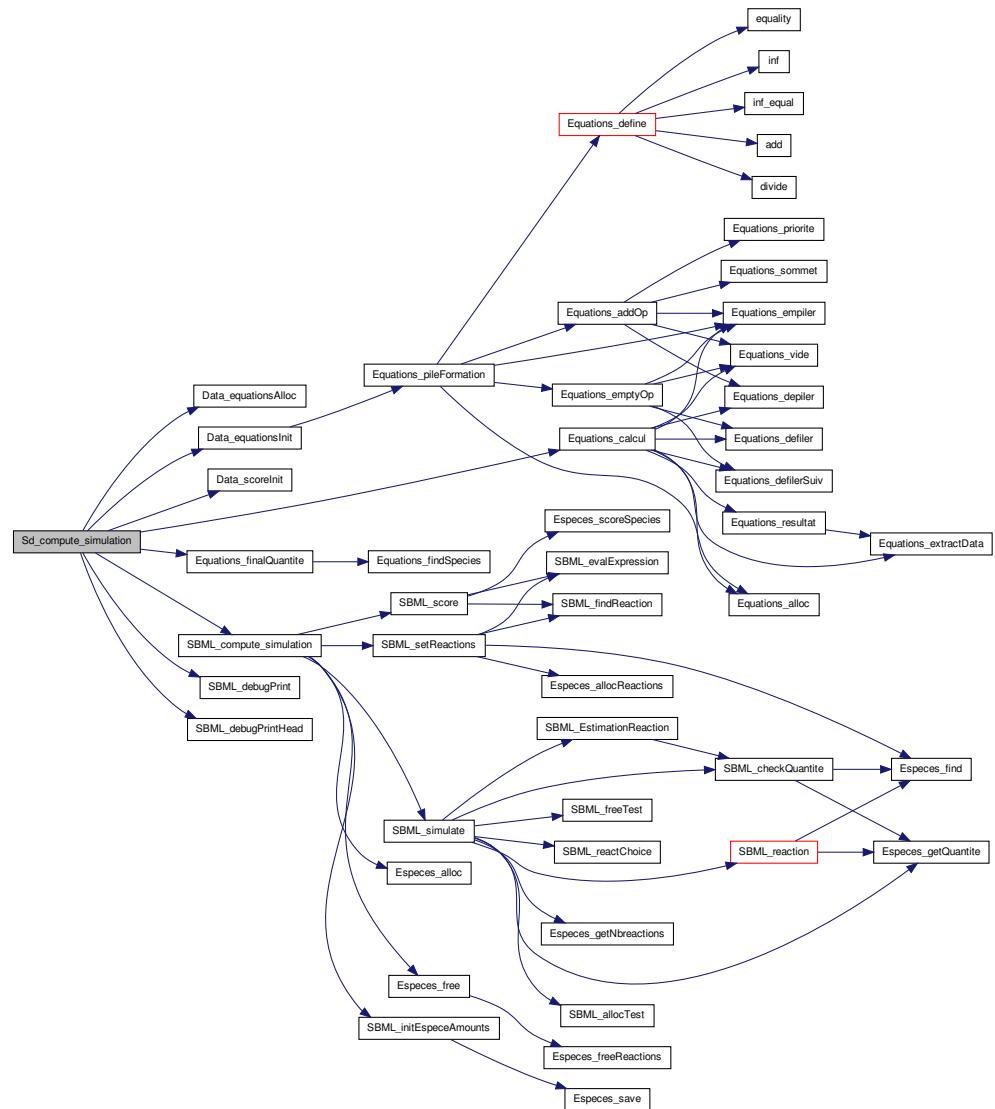
Energy value

Definition at line 56 of file `gsl_sd.c`.

References `ListParameters::banned`, `Data_equationsAlloc()`, `Data_equationsInit()`, `Data_scoreInit()`, `SimParameters::debugFile`, `Equations_calcul()`, `Equations_finalQuantite()`, `ListParameters::model`, `Score::name`, `ListParameters::nb_banned`, `ListParameters::nb_equations`, `Score::nb_species`, `SimParameters::out`, `SimParameters::pile`, `Score::quantite`, `SimParameters::r`, `SBML_compute_simulation()`, `SBML_debugPrint()`, `SBML_debugPrintHead()`, `Score::species`, `Score::species_amount`, `Score::species_weight`, `Score::taille`, `Score::tailleSpecies`, and `SimParameters::y`.

Referenced by `Sd_compute_standard_deviation()`.

Here is the call graph for this function:



**4.17.2.2 void Sd\_compute\_standard\_deviation ( pListParameters a, double \* fluxRatio, double \* result\_tab, char \*\* files\_path, int number\_arg, int debug )**

Compute the standard deviation.

#### Author

Amine Ghozlane

## Parameters

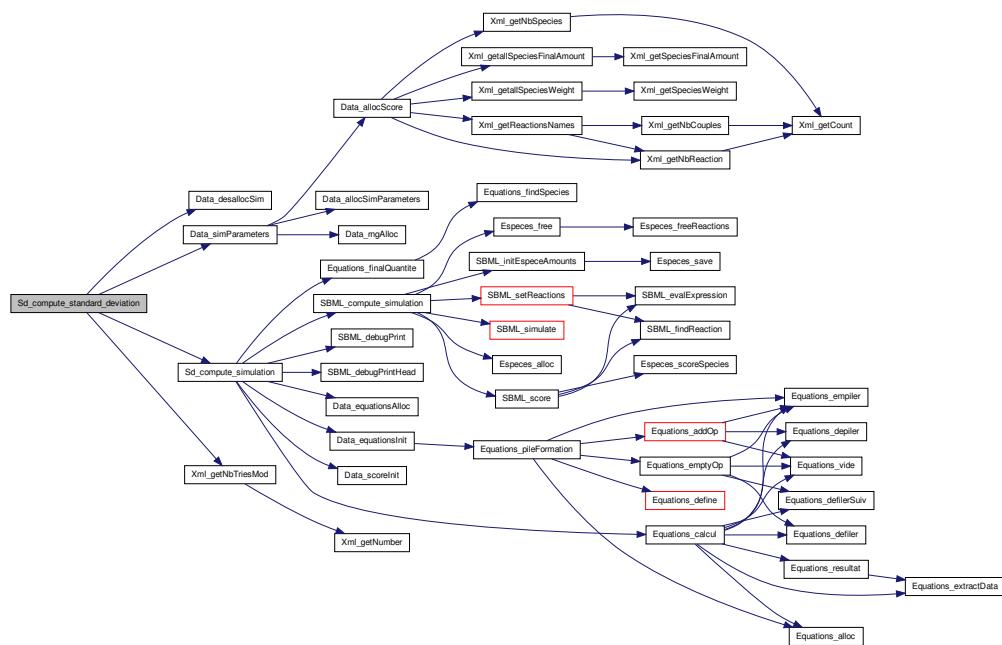
<i>a</i>	struct <a href="#">ListParameters</a>
<i>fluxRatio</i>	Ratio parameters
<i>result_tab</i>	Result table
<i>files_path</i>	List of paths
<i>number_arg</i>	Number of simulation
<i>debug</i>	Debug flag

Definition at line 106 of file `gsl_sd.c`.

References ListParameters::conf, Data\_desallocSim(), Data\_simParameters(), ListParameters::nb\_parameters, Sd\_compute\_simulation(), Xml\_getNbTriesMod(), and SimParameters::y.

Referenced by Mpi\_slave().

Here is the call graph for this function:



## 4.18 metaboflux/src/MetaBoFlux.c File Reference

## Program for simulated annealing analysis.

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

```
#include <string.h>
#include <assert.h>
#include <getopt.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_multimin.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_siman.h>
#include <gsl/gsl_ieee_utils.h>
#include <gsl/gsl_statistics_double.h>
#include <sbml/SBMLTypes.h>
#include <libxml/parser.h>
#include <libxml>xpath.h>
#include "mpi.h"
#include "especies.h"
#include "equations.h"
#include "simulation.h"
#include "xml_parameter.h"
#include "data_parameters.h"
#include "gsl_recuit.h"
#include "gsl_mod.h"
#include "gsl_min.h"
#include "gsl_sd.h"
#include "mpi_load.h"
```

Include dependency graph for MetaBoFlux.c:



## Data Structures

- struct [ListArguments](#)

*List of arguments.*

## Functions

- `pListArguments alloc_arguments (void)`  
*Alloc memory for the struct pListArguments.*
- `void free_arguments (pListArguments args)`  
*Free the struct pListArguments.*
- `void help_print (void)`  
*Print the usage help.*
- `void error_activity (int activity)`  
*Print the error notification.*
- `void argument_analysis (int argc, char **argv, pListArguments args)`  
*Print the error notification.*
- `void check_common (pListArguments args)`  
*Check obligatory arguments.*
- `void check_arguments (pListArguments args)`  
*Check program arguments.*
- `int main (int argc, char **argv)`  
*Enter in the program for simulated annealing.*

### 4.18.1 Detailed Description

Program for simulated annealing analysis. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/meta>)  
Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

{Amine Ghozlane}

#### Version

3.0

#### Date

9 novembre 2009

Definition in file [MetaBoFlux.c](#).

---

### 4.18.2 Function Documentation

#### 4.18.2.1 `pListArguments alloc_arguments ( void )`

Alloc memory for the struct pListArguments.

##### Author

Amine Ghozlane

Definition at line 103 of file MetaBoFlux.c.

References ListArguments::activity, ListArguments::debug, ListArguments::files\_path, ListArguments::group, and ListArguments::port.

Referenced by main().

#### 4.18.2.2 `void argument_analysis ( int argc, char ** argv, pListArguments args )`

Print the error notification.

##### Author

Amine Ghozlane

##### Parameters

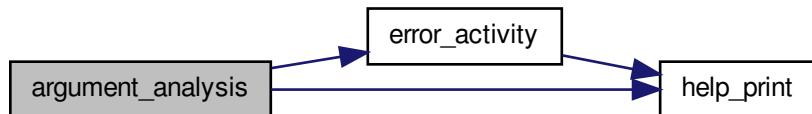
<code>argc</code>	Number of arguments
<code>argv</code>	List of arguments
<code>args</code>	struct <a href="#">ListArguments</a>

Definition at line 185 of file MetaBoFlux.c.

References ListArguments::activity, ListArguments::debug, error\_activity(), ListArguments::files\_path, ListArguments::group, help\_print(), and ListArguments::port.

Referenced by main().

Here is the call graph for this function:



#### 4.18.2.3 void check\_arguments ( pListArguments args )

Check program arguments.

##### Author

Amine Ghozlane

##### Parameters

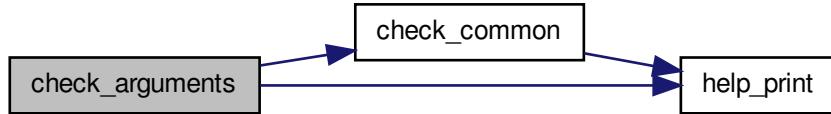
args	struct <a href="#">ListArguments</a>
------	--------------------------------------

Definition at line 361 of file MetaBoFlux.c.

References [ListArguments::activity](#), [check\\_common\(\)](#), [ListArguments::files\\_path](#), and [help\\_print\(\)](#).

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

Here is the call graph for this function:



#### 4.18.2.4 void check\_common ( pListArguments args )

Check obligatory arguments.

##### Author

Amine Ghozlane

##### Parameters

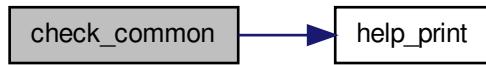
args	struct <a href="#">ListArguments</a>
------	--------------------------------------

Definition at line 332 of file MetaBoFlux.c.

References [ListArguments::files\\_path](#), and [help\\_print\(\)](#).

Referenced by [check\\_arguments\(\)](#).

Here is the call graph for this function:



#### 4.18.2.5 void error\_activity ( int *activity* )

Print the error notification.

##### Author

Amine Ghozlane

##### Parameters

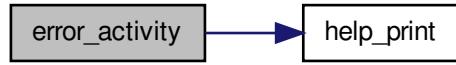
<i>activity</i>	Activity chosen
-----------------	-----------------

Definition at line 170 of file MetaBoFlux.c.

References `help_print()`.

Referenced by `argument_analysis()`.

Here is the call graph for this function:



#### 4.18.2.6 void free\_arguments ( pListArguments *arguments* )

Free the struct pListArguments.

##### Author

Amine Ghozlane

**Parameters**

<i>arguments</i>	struct pListArguments
------------------	-----------------------

Definition at line 131 of file MetaBoFlux.c.

References ListArguments::files\_path.

Referenced by main().

**4.18.2.7 void help\_print( void )**

Print the usage help.

**Author**

Amine Ghozlane

Definition at line 142 of file MetaBoFlux.c.

Referenced by argument\_analysis(), check\_arguments(), check\_common(), and error\_activity().

**4.18.2.8 int main( int argc, char \*\* argv )**

Enter in the program for simulated annealing.

**Author**

Amine Ghozlane

**Parameters**

<i>argc</i>	Number of arguments
<i>argv</i>	List of arguments

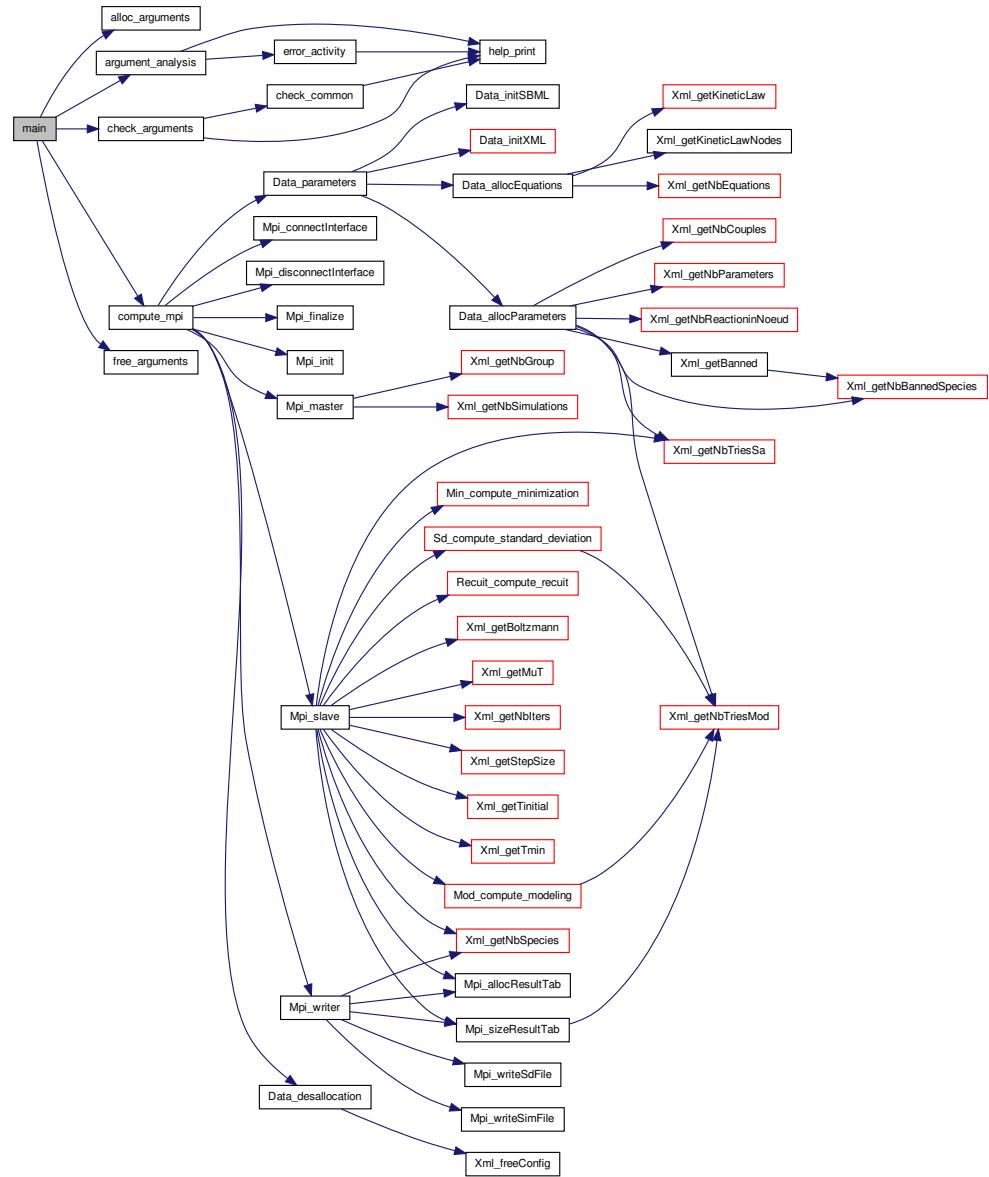
**Returns**

EXIT\_SUCCESS Normal stop of the program

Definition at line 399 of file MetaBoFlux.c.

References ListArguments::activity, alloc\_arguments(), argument\_analysis(), check\_arguments(), compute\_mpi(), ListArguments::debug, ListArguments::files\_path, free\_arguments(), ListArguments::group, and ListArguments::port.

Here is the call graph for this function:



#### 4.19 metaboflux/src/mpi\_load.c File Reference

Parallelize the program.

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

```
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <assert.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_multimin.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_siman.h>
#include <gsl/gsl_ieee_utils.h>
#include <gsl/gsl_statistics_double.h>
#include <sbml/SBMLTypes.h>
#include <libxml/parser.h>
#include <libxml>xpath.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include "xml_parameter.h"
#include "especies.h"
#include "equations.h"
#include "simulation.h"
#include "data_parameters.h"
#include "gsl_recuit.h"
#include "gsl_mod.h"
#include "gsl_min.h"
#include "gsl_sd.h"
#include "mpi.h"
#include "mpi_load.h"
```

Include dependency graph for mpi\_load.c:



## Functions

- int **Mpi\_connectInterface** (int port)
 

*Enter in the program for standard deviation.*
- void **Mpi\_disconnectInterface** (int desc)
 

*Disconnection to the interface.*
- void **Mpi\_init** (int argc, char \*\*argv, int \*tab)
 

*Enter in the program for standard deviation.*
- void **Mpi\_master** (pListParameters allone, char \*\*files\_path, int activity, int group, int myid, int numprocs, int desc)
 

*Master process.*
- void **Mpi\_writer** (pListParameters allone, char \*\*files\_path, int activity, int group, int myid)
 

*Master process.*
- int **Mpi\_sizeResultTab** (pListParameters allone, int activity, int group, int nb\_species)
 

*Determine the size of the result tab.*
- double \* **Mpi\_allocResultTab** (int tailleTab)
 

*Allocate the result tab.*
- void **Mpi\_writeSimFile** (pListParameters allone, FILE \*out, FILE \*logOut, double \*result\_tab, int group, int tailleSpecies, int nb\_species)
 

*Determine the size of the result tab.*
- void **Mpi\_writeSdFile** (FILE \*out, double \*result\_tab, int tailleTab)
 

*Determine the size of the result tab.*
- void **Mpi\_slave** (pListParameters allone, char \*\*files\_path, int activity, int group, int debug)
- void **Mpi\_finalize** (void)
 

*End MPI execution environment.*
- void **compute\_mpi** (int argc, char \*\*argv, char \*\*files\_path, int activity, int group, int debug, int port)
 

*Compute the simulated annealing through mpi.*

### 4.19.1 Detailed Description

Parallelize the program. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metabof>)  
 Copyright (C) 2010 Amine Ghazlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

{Amine Ghozlane}

**Version**

2.0

**Date**

9 novembre 2009

Definition in file [mpi\\_load.c](#).

#### 4.19.2 Function Documentation

**4.19.2.1 void compute\_mpi ( int *argc*, char \*\* *argv*, char \*\* *files\_path*, int *activity*, int *group*, int *debug*, int *port* )**

Compute the simulated annealing through mpi.

**Author**

Amine Ghozlane

**Parameters**

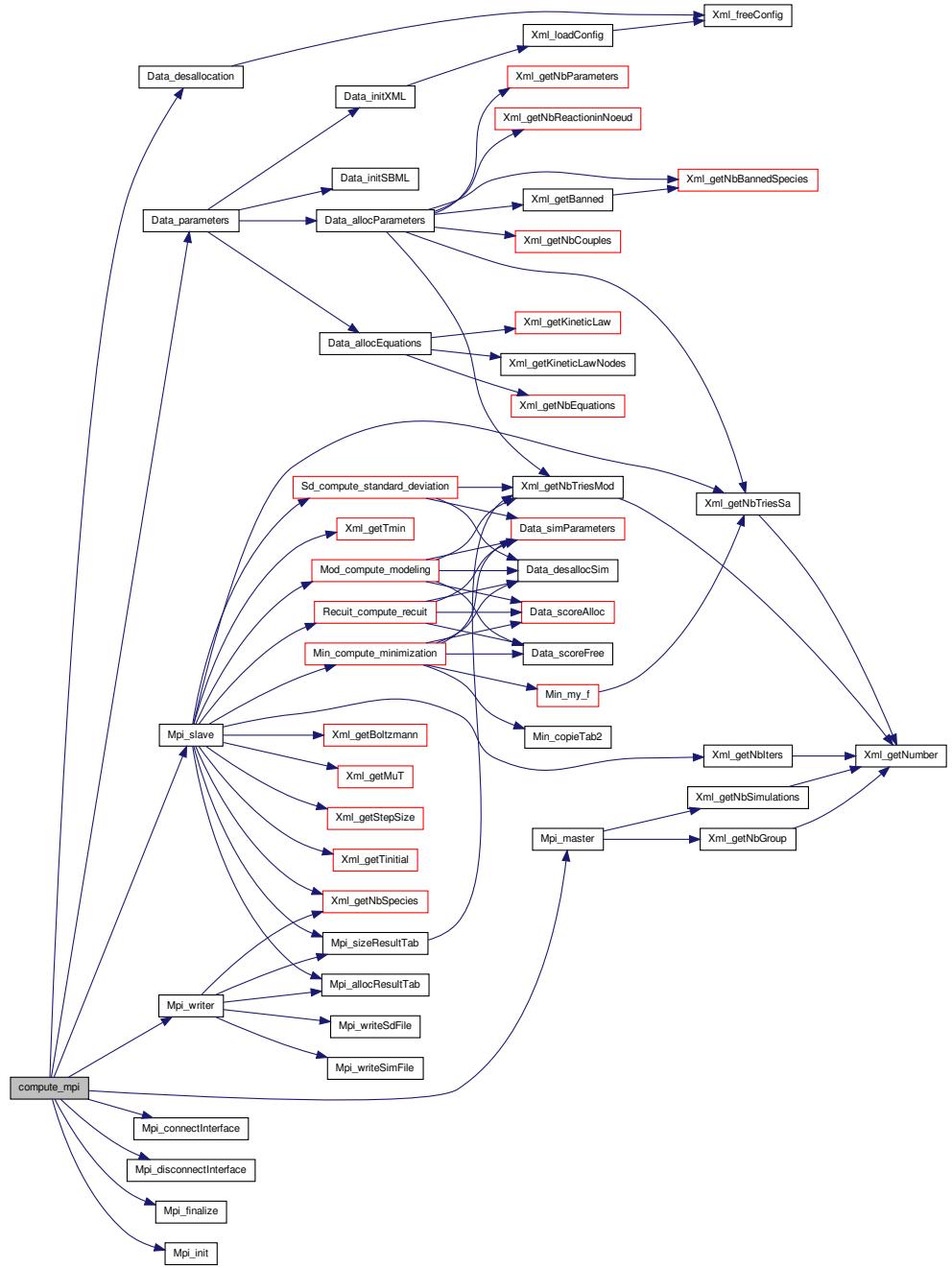
<i>argc</i>	Number of arguments
<i>argv</i>	List of arguments
<i>files_path</i>	List of paths
<i>activity</i>	Chosen activity
<i>group</i>	Group flag
<i>debug</i>	Debug flag
<i>port</i>	Interface port

Definition at line 546 of file [mpi\\_load.c](#).

References [Data\\_desallocation\(\)](#), [Data\\_parameters\(\)](#), [Mpi\\_connectInterface\(\)](#), [Mpi\\_disconnectInterface\(\)](#), [Mpi\\_finalize\(\)](#), [Mpi\\_init\(\)](#), [Mpi\\_master\(\)](#), [Mpi\\_slave\(\)](#), and [Mpi\\_writer\(\)](#).

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

Here is the call graph for this function:



**4.19.2.2 double \* Mpi\_allocResultTab ( int *tailleTab* )**

Allocate the result tab.

**Author**

Amine Ghozlane

**Parameters**

<i>tailleTab</i>	Size of Result tab
------------------	--------------------

**Returns**

Address of the allocated space

Definition at line 355 of file mpi\_load.c.

Referenced by Mpi\_slave(), and Mpi\_writer().

**4.19.2.3 int Mpi\_connectInterface ( int *port* )**

Enter in the program for standard deviation.

**Author**

Amine Ghozlane

**Parameters**

<i>port</i>	Connection port
-------------	-----------------

**Returns**

Socket

Definition at line 69 of file mpi\_load.c.

Referenced by compute\_mpi().

**4.19.2.4 void Mpi\_disconnectInterface ( int *desc* )**

Disconnection to the interface.

**Author**

Amine Ghozlane

**Parameters**

<i>desc</i>	Socket
-------------	--------

Definition at line 109 of file mpi\_load.c.

Referenced by compute\_mpi().

#### 4.19.2.5 void Mpi\_finalize ( void )

End MPI execution environment.

##### Author

Amine Ghozlane

Definition at line 527 of file mpi\_load.c.

Referenced by compute\_mpi().

#### 4.19.2.6 void Mpi\_init ( int argc, char \*\* argv, int \* tab )

Enter in the program for standard deviation.

##### Author

Amine Ghozlane

##### Parameters

<i>argc</i>	Number of arguments
<i>argv</i>	List of arguments
<i>tab</i>	Table

Definition at line 123 of file mpi\_load.c.

Referenced by compute\_mpi().

#### 4.19.2.7 void Mpi\_master ( pListParameters allone, char \*\* files\_path, int activity, int group, int myid, int numprocs, int desc )

Master process.

##### Author

Amine Ghozlane

##### Parameters

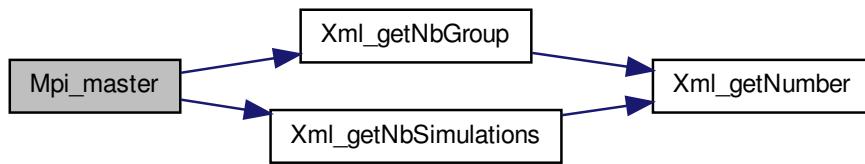
<i>allone</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>files_path</i>	List of paths
<i>activity</i>	Chosen activity
<i>group</i>	Group flag
<i>myid</i>	Id of the thread
<i>numprocs</i>	Number of thread
<i>desc</i>	Socket

Definition at line 154 of file mpi\_load.c.

References ListParameters::conf, ListParameters::nb\_parameters, Xml\_getNbGroup(), and Xml\_getNbSimulations().

Referenced by compute\_mpi().

Here is the call graph for this function:



#### 4.19.2.8 int Mpi\_sizeResultTab ( pListParameters *alone*, int *activity*, int *group*, int *nb\_species* )

Determine the size of the result tab.

##### Author

Amine Ghozlane

##### Parameters

<i>alone</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>activity</i>	Chosen activity
<i>group</i>	Group flag
<i>nb_species</i>	Number of interest species

##### Returns

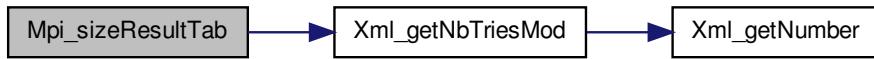
Size of Result tab

Definition at line 334 of file mpi\_load.c.

References ListParameters::conf, ListParameters::model, ListParameters::nb\_parameters, and Xml\_getNbTriesMod().

Referenced by Mpi\_slave(), and Mpi\_writer().

Here is the call graph for this function:



**4.19.2.9 void Mpi\_slave ( pListParameters *allone*, char \*\* *files\_path*, int *activity*, int *group*, int *debug* )**

#### Author

Amine Ghozlane

#### Parameters

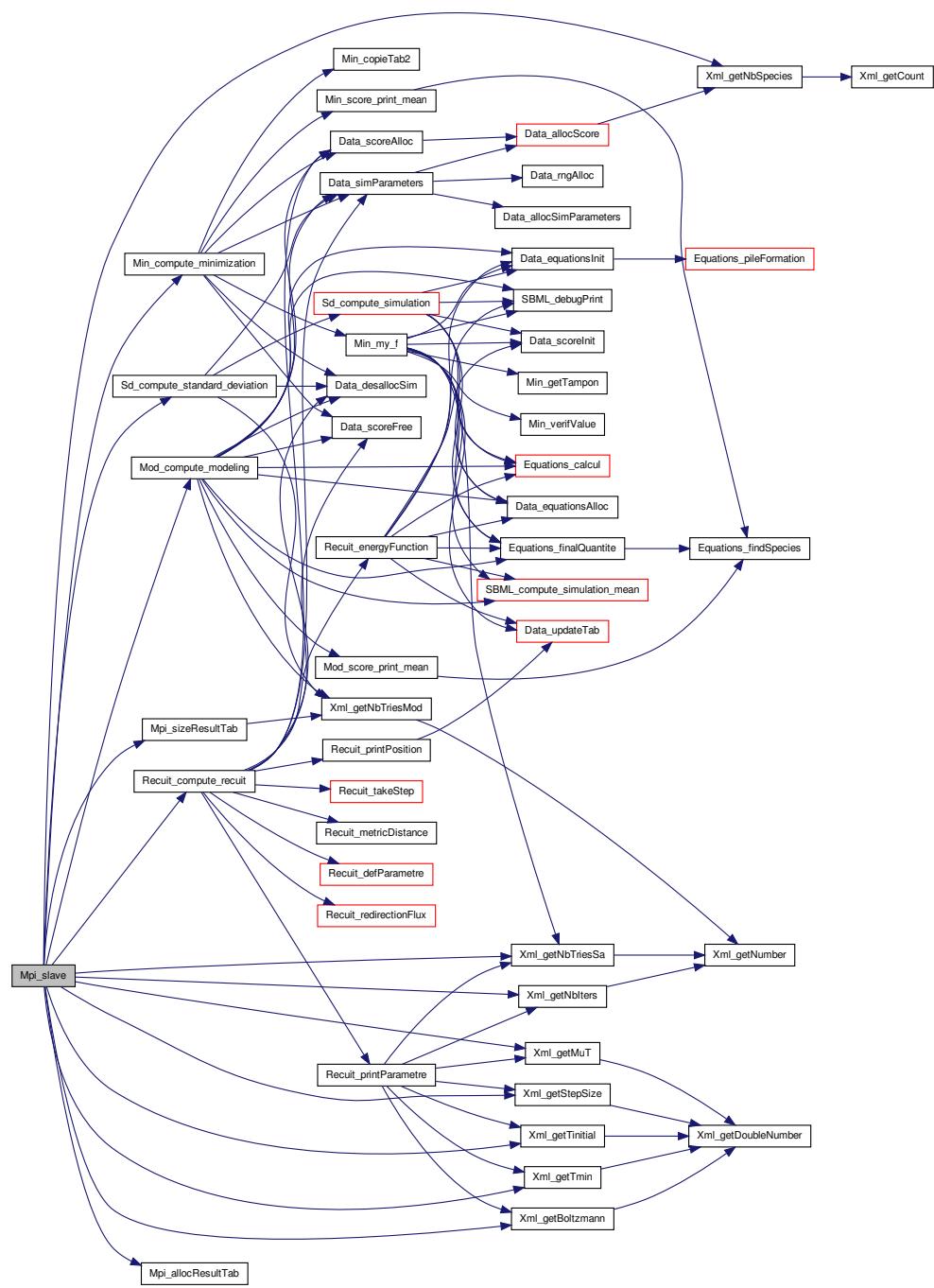
<i>allone</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>files_path</i>	List of paths
<i>activity</i>	Activity chosen
<i>group</i>	Group flag
<i>debug</i>	Debug flag

Definition at line 440 of file `mpi_load.c`.

References `ListParameters::conf`, `Min_compute_minimization()`, `Mod_compute_modeling()`, `Mpi_allocResultTab()`, `Mpi_sizeResultTab()`, `ListParameters::nb_parameters`, `Recuit_compute_recuit()`, `Sd_compute_standard_deviation()`, `Xml_getBoltzmann()`, `Xml_getMuT()`, `Xml_getNbIter()`, `Xml_getNbSpecies()`, `Xml_getNbTriesSa()`, `Xml_getStepSize()`, `Xml_getTinitial()`, and `Xml_getTmin()`.

Referenced by `compute_mpi()`.

Here is the call graph for this function:



4.19.2.10 void Mpi\_writer ( pListParameters *alone*, char \*\* *files\_path*, int *activity*, int *group*, int *myid* )

Master process.

#### Author

Amine Ghozlane

#### Parameters

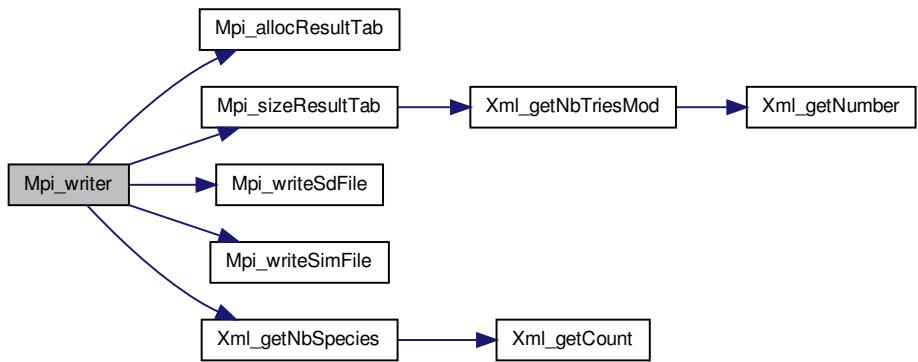
<i>alone</i>	Global parameters : struct ListParameters
<i>files_path</i>	List of paths
<i>activity</i>	Chosen activity
<i>group</i>	Group flag
<i>myid</i>	Id of the thread

Definition at line 273 of file mpi\_load.c.

References ListParameters::conf, ListParameters::model, Mpi\_allocResultTab(), Mpi\_sizeResultTab(), Mpi\_writeSdFile(), Mpi\_writeSimFile(), and Xml\_getNbSpecies().

Referenced by compute\_mpi().

Here is the call graph for this function:



4.19.2.11 void Mpi\_writeSdFile ( FILE \* *out*, double \* *result\_tab*, int *tailleTab* )

Determine the size of the result tab.

#### Author

Amine Ghozlane

**Parameters**

<i>out</i>	Result file
<i>result_tab</i>	Result table
<i>tailleTab</i>	Size of result table

Definition at line 421 of file mpi\_load.c.

Referenced by Mpi\_writer().

**4.19.2.12 void Mpi\_writeSimFile ( pListParameters *allone*, FILE \* *out*, FILE \* *logOut*, double \* *result\_tab*, int *group*, int *tailleSpecies*, int *nb\_species* )**

Determine the size of the result tab.

**Author**

Amine Ghozlane

**Parameters**

<i>allone</i>	Global parameters : struct <a href="#">ListParameters</a>
<i>out</i>	Result file
<i>logOut</i>	Log file
<i>result_tab</i>	Result table
<i>group</i>	Group flag
<i>tailleSpecies</i>	Number of species
<i>nb_species</i>	Number of interest species

Definition at line 375 of file mpi\_load.c.

References ListParameters::nb\_parameters.

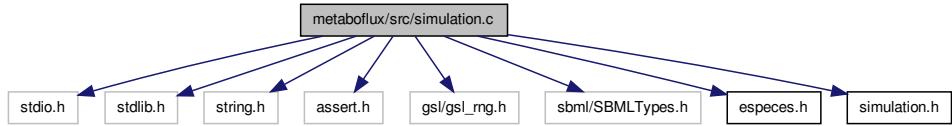
Referenced by Mpi\_writer().

## 4.20 metaboflux/src/simulation.c File Reference

Simulate a petri net.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <gsl/gsl_rng.h>
#include <sbml/SBMLTypes.h>
#include "especies.h"
#include "simulation.h"
```

Include dependency graph for simulation.c:



## Functions

- void **SBML\_initEspecieAmounts** (Model\_t \*mod, pEspecies molecules, int nbE-species)
 

*Alloc memory and initialize the struct Especies.*
- int **SBML\_findReaction** (char \*\*reaction, const char \*react, int nb\_reaction)
 

*Determine if the reaction is study.*
- double **SBML\_evalExpression** (const char \*formule)
 

*Get the reaction ratio define in the sbml.*
- void **SBML\_setReactions** (Model\_t \*mod, pEspecies molecules, pScore result, double \*reactions\_ratio, int nbReactions, int nbEspecies)
 

*Alloc memory and initialize the struct Especies.*
- int **SBML\_checkQuantite** (Model\_t \*mod, Reaction\_t \*react, int nbEspecies, pE-species molecules)
 

*Determine the number of reaction for one molecule.*
- Reaction\_t \* **SBML\_reactChoice** (pEspecies molecules, const gsl\_rng \*r, int ref)
 

*Determine randomly the reaction to achieve for several nodes reactions.*
- void **SBML\_reaction** (Model\_t \*mod, pEspecies molecules, Reaction\_t \*react, int nbEspecies)
 

*Simulation of a discrete transision.*
- void **SBML\_allocTest** (pTestReaction T, int nbReactions)
 

*Alloc memory and initialize the struct pTestReaction.*
- void **SBML\_freeTest** (pTestReaction T)
 

*Free memory of the struct TestReaction.*
- int **SBML\_EstimationReaction** (Model\_t \*mod, pTestReaction T, pEspecies molecules, int ref, int nbEspecies)
 

*Alloc memory and initialize the struct Especies.*
- int **SBML\_simulate** (Model\_t \*mod, pEspecies molecules, const gsl\_rng \*r, pTestReaction T, char \*\*banned, int nbBanned, int nbEspecies, int ref)
 

*Simulate one step of petri net.*
- void **SBML\_score** (Model\_t \*mod, pEspecies molecules, pScore result, double \*reactions\_ratio, int nbReactions, int nbEspecies)

*Alloc memory and initialize the struct `Especes`.*

- void `SBML_debugPrintHead` (FILE \*debugFile, int taille, char \*\*name)  
*Print the head of the debug file.*
- void `SBML_debugPrint` (FILE \*debugFile, int tailleSpecies, int taille, double \*quantite, double result)  
*Print the debuf file.*
- void `SBML_compute_simulation` (pScore result, Model\_t \*mod, double \*reactions\_ratio, gsl\_rng \*r, char \*\*banned, int nbBanned)  
*Simulation of metabolic network.*
- void `SBML_score_add` (pScore result, pScore result\_temp, FILE \*debugFile)  
*Add scores.*
- void `SBML_score_mean` (pScore result, int n)  
*Mean quantities for score.*
- void `SBML_compute_simulation_mean` (FILE \*debugFile, pScore result, pScore result\_temp, Model\_t \*mod, double \*reactions\_ratio, gsl\_rng \*r, char \*\*banned, int nbBanned, int nb\_simulation)  
*X time simulation of metabolic network.*

#### 4.20.1 Detailed Description

Simulate a petri net. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflux/>)  
Copyright (C) 2010 Amine Ghozlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

##### Author

{Amine Ghozlane}

##### Version

2.0

##### Date

27 octobre 2009

Definition in file [simulation.c](#).

## 4.20.2 Function Documentation

### 4.20.2.1 void SBML\_allocTest( pTestReaction *T*, int *nbReactions* )

Alloc memory and initialize the struct pTestReaction.

#### Author

Amine Ghozlane

#### Parameters

<i>T</i>	Empty struct <a href="#">TestReaction</a>
<i>nbReactions</i>	Number of reactions

Definition at line 279 of file simulation.c.

References [TestReaction::minStepTab](#), and [TestReaction::tabReactions](#).

Referenced by [SBML\\_simulate\(\)](#).

### 4.20.2.2 int SBML\_checkQuantite( Model\_t \* *mod*, Reaction\_t \* *react*, int *nbEspecies*, pEspecies *molecules* )

Determine the number of reaction for one molecule.

#### Author

Amine Ghozlane

#### Parameters

<i>mod</i>	Model of the SBML file
<i>react</i>	<a href="#">Reaction</a> id
<i>nbEspecies</i>	Number of molecules
<i>molecules</i>	Struct <a href="#">Especies</a>

#### Returns

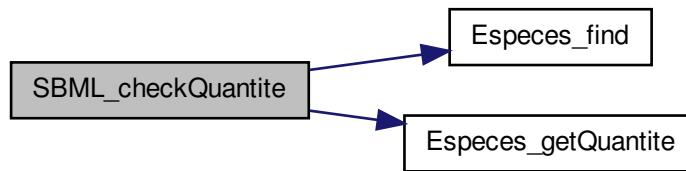
Number of reaction for one molecule

Definition at line 158 of file simulation.c.

References [Especies\\_find\(\)](#), and [Especies\\_getQuantite\(\)](#).

Referenced by [SBML\\_EstimationReaction\(\)](#), and [SBML\\_simulate\(\)](#).

Here is the call graph for this function:



**4.20.2.3 void SBML\_compute\_simulation ( pScore *result*, Model.t \* *mod*, double \* *reactions\_ratio*, gsl\_rng \* *r*, char \*\* *banned*, int *nbBanned* )**

Simulation of metabolic network.

#### Author

Amine Ghozlane

#### Parameters

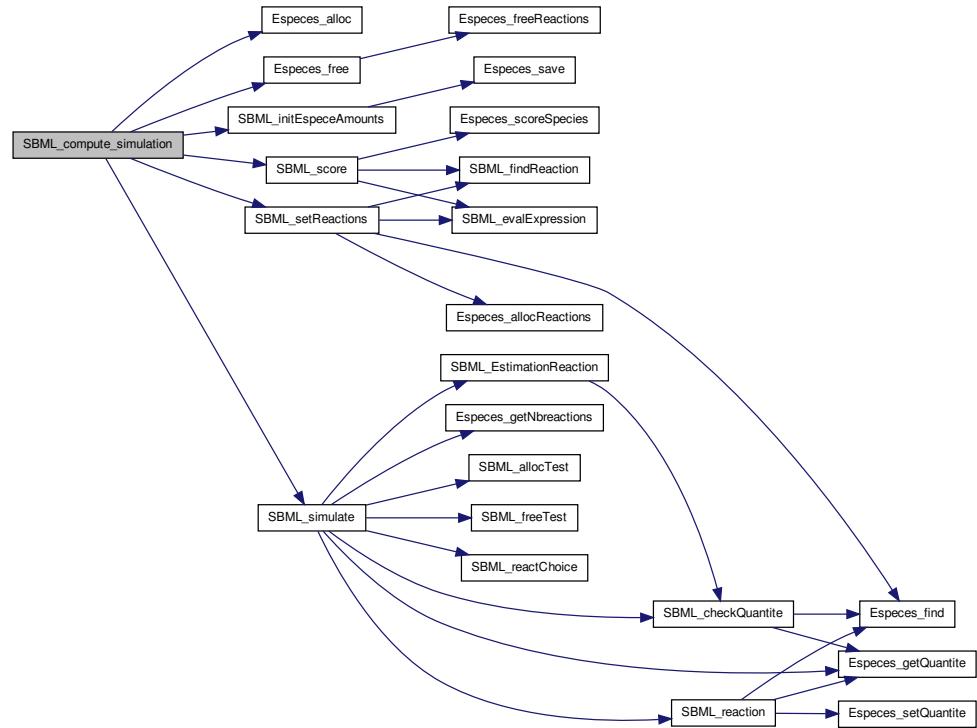
<i>result</i>	Struct <a href="#">Score</a>
<i>mod</i>	Model of the SBML file
<i>reactions_ratio</i>	List of computed reaction ratio
<i>r</i>	Random number generator
<i>banned</i>	List of banned compound
<i>nbBanned</i>	Number of banned compound

Definition at line 515 of file simulation.c.

References [Especies\\_alloc\(\)](#), [Especies\\_free\(\)](#), [SBML\\_initEspeceAmounts\(\)](#), [SBML\\_score\(\)](#), [SBML\\_setReactions\(\)](#), [SBML\\_simulate\(\)](#), [Score::tailleReactions](#), and [Score::tailleSpecies](#).

Referenced by [Sd\\_compute\\_simulation\(\)](#).

Here is the call graph for this function:



**4.20.2.4 void SBML\_compute\_simulation\_mean ( FILE \* debugFile, pScore result, pScore result\_temp, Model.t \* mod, double \* reactions\_ratio, gsl\_rng \* r, char \*\* banned, int nbBanned, int nb\_simulation )**

X time simulation of metabolic network.

#### Author

Amine Ghozlane

#### Parameters

<i>debugFile</i>	File use for debug
<i>result</i>	Struct <a href="#">Score</a> used for all the simulation
<i>result_temp</i>	Struct <a href="#">Score</a> used at each simulation step
<i>mod</i>	Model of the SBML file
<i>reactions_- ratio</i>	List of computed reaction ratio
<i>r</i>	Random number generator

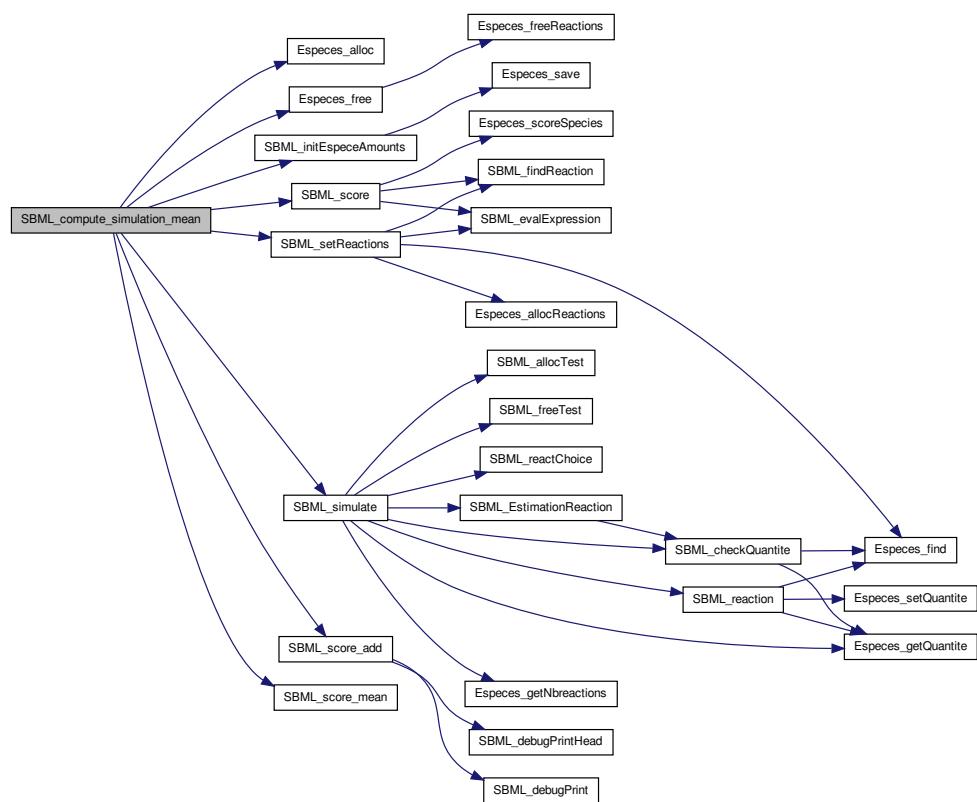
<i>banned</i>	List of banned compound
<i>nbBanned</i>	Number of banned compound
<i>nb_ - simulation</i>	Number of simulation step

Definition at line 614 of file simulation.c.

References Especies\_alloc(), Especies\_free(), SBML\_initEspeceAmounts(), SBML\_score(), SBML\_score\_add(), SBML\_score\_mean(), SBML\_setReactions(), and SBML\_simulate().

Referenced by Min\_my\_f(), Mod\_compute\_modeling(), and Recuit\_energyFunction().

Here is the call graph for this function:



4.20.2.5 void SBML\_debugPrint ( FILE \* *debugFile*, int *tailleSpecies*, int *taille*, double \* *quantite*, double *result* )

Print the debuf file.

**Author**

Amine Ghozlane

**Parameters**

<i>debugFile</i>	File use for debug
<i>tailleSpecies</i>	Number of molecules
<i>taille</i>	Number of molecules/reactions
<i>quantite</i>	Quantity of molecules/reactions
<i>result</i>	Energy value

Definition at line 486 of file simulation.c.

Referenced by Min\_my\_f(), Mod\_compute\_modeling(), Recuit\_energyFunction(), SBML\_-score\_add(), and Sd\_compute\_simulation().

**4.20.2.6 void SBML\_debugPrintHead ( FILE \* *debugFile*, int *taille*, char \*\* *name* )**

Print the head of the debug file.

**Author**

Amine Ghozlane

**Parameters**

<i>debugFile</i>	File use for debug
<i>taille</i>	Number of molecules/reactions
<i>name</i>	List of molecules/reactions

Definition at line 465 of file simulation.c.

Referenced by SBML\_score\_add(), and Sd\_compute\_simulation().

**4.20.2.7 int SBML\_EstimationReaction ( Model\_t \* *mod*, pTestReaction *T*, pEspecies *molecules*, int *ref*, int *nbEspecies* )**

Alloc memory and initialize the struct [Especies](#).

**Author**

Amine Ghozlane

**Parameters**

<i>mod</i>	Model of the SBML file
<i>T</i>	Struct <a href="#">TestReaction</a> gives data on reaction
<i>molecules</i>	Struct <a href="#">Especies</a>
<i>ref</i>	Number reference of one molecule
<i>nbEspecies</i>	Number of molecules

**Returns**

Estimated number of feasible step by reaction

Definition at line 322 of file simulation.c.

References Reaction::link, TestReaction::minStepTab, SBML\_checkQuantite(), Reaction::suivant, Espesee::system, and TestReaction::tabReactions.

Referenced by SBML\_simulate().

Here is the call graph for this function:

**4.20.2.8 double SBML\_evalExpression ( const char \* *formule* )**

Get the reaction ratio define in the sbml.

**Author**

Amine Ghozlane

**Parameters**

<i>formule</i>	Formule SBML
----------------	--------------

**Returns**

Return double value of the constraint

Definition at line 88 of file simulation.c.

Referenced by SBML\_score(), and SBML\_setReactions().

**4.20.2.9 int SBML\_findReaction ( char \*\* *reaction*, const char \* *react*, int *nb\_reaction* )**

Determine if the reaction is study.

**Author**

Amine Ghozlane

**Parameters**

<i>reaction</i>	List of reactions
<i>react</i>	<a href="#">Reaction</a> of interest
<i>nb_reaction</i>	Number of reactions

**Returns**

Number of the molecules if it's study

Definition at line 69 of file simulation.c.

Referenced by [SBML\\_score\(\)](#), and [SBML\\_setReactions\(\)](#).

**4.20.2.10 void SBML\_freeTest( pTestReaction *T* )**

Free memory of the struct [TestReaction](#).

**Author**

Amine Ghozlane

**Parameters**

<i>T</i>	Struct <a href="#">TestReaction</a> gives data on reaction
----------	--

Definition at line 304 of file simulation.c.

References [TestReaction::minStepTab](#), and [TestReaction::tabReactions](#).

Referenced by [SBML\\_simulate\(\)](#).

**4.20.2.11 void SBML\_initEspecieAmounts( Model\_t \* *mod*, pEspecies *molecules*, int *nbEspecies* )**

Alloc memory and initialize the struct [Especies](#).

**Author**

Amine Ghozlane

**Parameters**

<i>mod</i>	Model of the SBML file
<i>molecules</i>	Struct <a href="#">Especies</a>
<i>nbEspecies</i>	Number of molecules

Definition at line 46 of file simulation.c.

References [Especies\\_save\(\)](#).

Referenced by [SBML\\_compute\\_simulation\(\)](#), and [SBML\\_compute\\_simulation\\_mean\(\)](#).

Here is the call graph for this function:



#### 4.20.2.12 Reaction\_t \* SBML\_reactChoice ( pEspecies molecules, const gsl\_rng \* r, int ref )

Determine randomly the reaction to achieve for several nodes reactions.

##### Author

Amine Ghozlane

##### Parameters

<i>molecules</i>	Struct <a href="#">Especies</a>
<i>r</i>	Random number generator
<i>ref</i>	Number reference of one molecule

##### Returns

Id of the selected reaction

Definition at line 206 of file simulation.c.

References Reaction::link, Reaction::suivant, and Especies::system.

Referenced by SBML\_simulate().

#### 4.20.2.13 void SBML\_reaction ( Model\_t \* mod, pEspecies molecules, Reaction\_t \* react, int nbEspecies )

Simulation of a discrete transition.

##### Author

Amine Ghozlane

##### Parameters

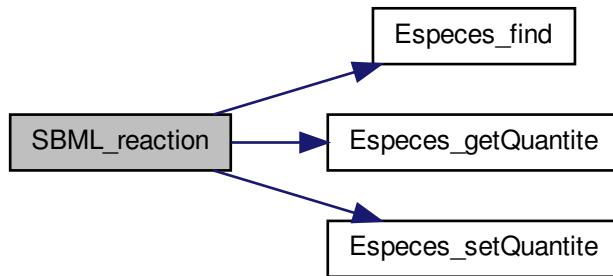
<i>mod</i>	Model of the SBML file
<i>molecules</i>	Struct <a href="#">Especies</a>
<i>react</i>	Reaction id
<i>nbEspecies</i>	Number of molecules

Definition at line 244 of file simulation.c.

References `Especies_find()`, `Especies_getQuantite()`, and `Especies_setQuantite()`.

Referenced by `SBML_simulate()`.

Here is the call graph for this function:



**4.20.2.14 void SBML\_score ( Model.t \* mod, pEspecies molecules, pScore result, double \* reactions\_ratio, int nbReactions, int nbEspecies )**

Alloc memory and initialize the struct `Especies`.

#### Author

Amine Ghozlane

#### Parameters

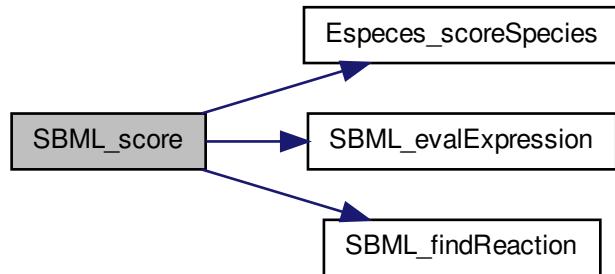
<code>mod</code>	Model of the SBML file
<code>molecules</code>	Struct <code>Especies</code>
<code>result</code>	Struct <code>Score</code>
<code>reactions_- ratio</code>	List of computed reaction ratio
<code>nbReactions</code>	Number of reactions
<code>nbEspecies</code>	Number of molecules

Definition at line 420 of file simulation.c.

References `Especies_scoreSpecies()`, `Score::name`, `Score::nb_reaction`, `Score::quantite`, `Score::reaction`, `SBML_evalExpression()`, and `SBML_findReaction()`.

Referenced by `SBML_compute_simulation()`, and `SBML_compute_simulation_mean()`.

Here is the call graph for this function:



#### 4.20.2.15 void SBML\_score\_add( pScore result, pScore result\_temp, FILE \* debugFile )

Add scores.

##### Author

Amine Ghozlane

##### Parameters

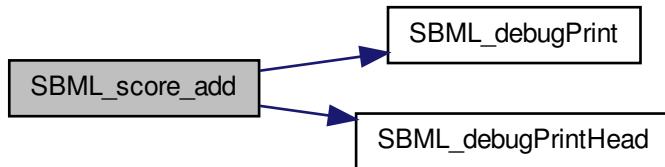
<i>result</i>	Struct <a href="#">Score</a> used for all the simulation
<i>result_temp</i>	Struct <a href="#">Score</a> used at each simulation step
<i>debugFile</i>	File use for debug

Definition at line 560 of file simulation.c.

References Score::name, Score::quantite, SBML\_debugPrint(), SBML\_debugPrintHead(), Score::taille, and Score::tailleSpecies.

Referenced by SBML\_compute\_simulation\_mean().

Here is the call graph for this function:



#### 4.20.2.16 void SBML\_score\_mean ( pScore result, int n )

Mean quantities for score.

##### Author

Amine Ghozlane

##### Parameters

<i>result</i>	Struct <a href="#">Score</a>
<i>n</i>	Number of simulation step

Definition at line 591 of file simulation.c.

References Score::quantite, and Score::taille.

Referenced by SBML\_compute\_simulation\_mean().

#### 4.20.2.17 void SBML\_setReactions ( Model\_t \* mod, pEspecies molecules, pScore result, double \* reactions\_ratio, int nbReactions, int nbEspecies )

Alloc memory and initialize the struct [Especies](#).

##### Author

Amine Ghozlane

##### Parameters

<i>mod</i>	Model of the SBML file
<i>molecules</i>	Struct <a href="#">Especies</a>
<i>result</i>	Struct <a href="#">Score</a>
<i>reactions_ratio</i>	List of computed reaction ratio

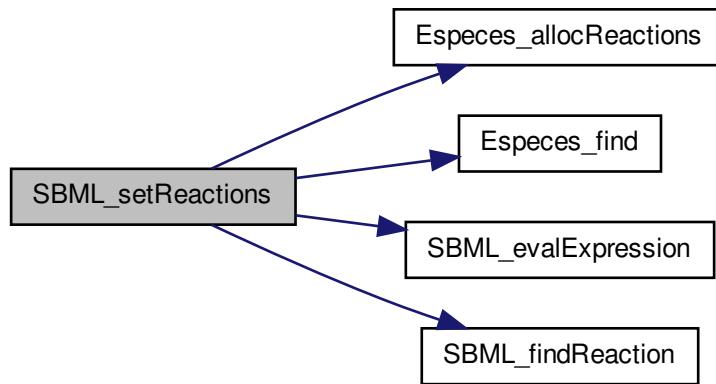
<code>nbReactions</code>	Number of reaction
<code>nbEspecies</code>	Number of molecules

Definition at line 105 of file simulation.c.

References `Especies_allocReactions()`, `Especies_find()`, `Score::nb_reaction`, `Score::reaction`, `SBML_evalExpression()`, and `SBML_findReaction()`.

Referenced by `SBML_compute_simulation()`, and `SBML_compute_simulation_mean()`.

Here is the call graph for this function:



**4.20.2.18** `int SBML_simulate ( Model_t * mod, pEspecies molecules, const gsl_rng * r, pTestReaction T, char ** banned, int nbBanned, int nbEspecies, int ref )`

Simulate one step of petri net.

#### Author

Amine Ghozlane

#### Parameters

<code>mod</code>	Model of the SBML file
<code>molecules</code>	Struct <code>Especies</code>
<code>r</code>	Random number generator
<code>T</code>	Struct <code>TestReaction</code> gives data on reaction
<code>banned</code>	List of banned compound
<code>nbBanned</code>	Number of banned compound

<code>nbEspecies</code>	Number of molecules
<code>ref</code>	Number reference of one molecule

**Returns**

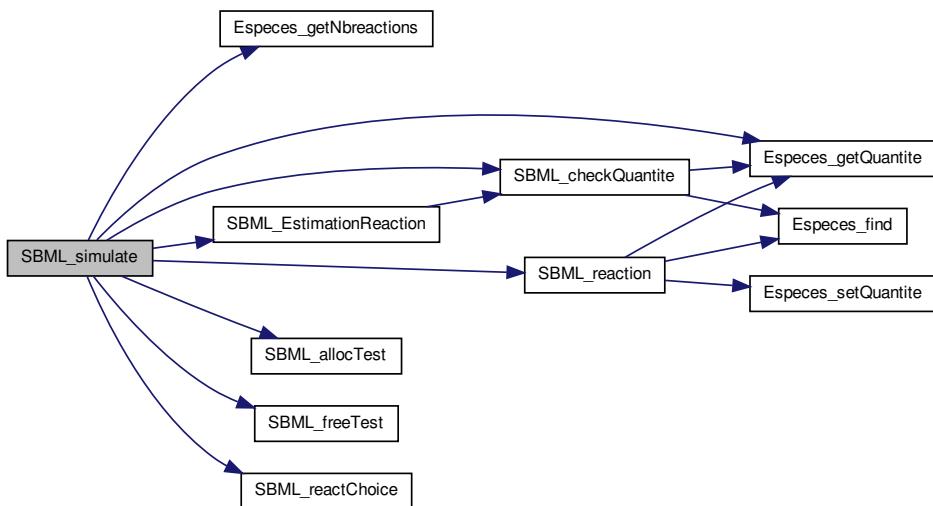
Condition of stop/pursue

Definition at line 356 of file simulation.c.

References `Especies_getNbreactions()`, `Especies_getQuantite()`, `Reaction::link`, `SBML_allocTest()`, `SBML_checkQuantite()`, `SBML_EstimationReaction()`, `SBML_freeTest()`, `SBML_reactChoice()`, `SBML_reaction()`, and `Especies::system`.

Referenced by `SBML_compute_simulation()`, and `SBML_compute_simulation_mean()`.

Here is the call graph for this function:



## 4.21 metaboflux/src/xml\_parameter.c File Reference

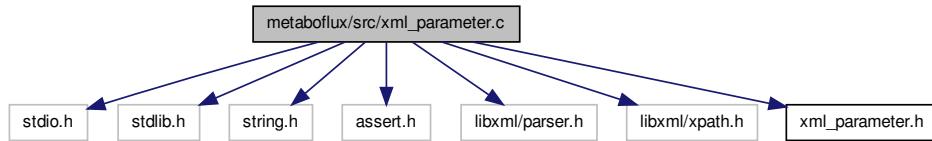
Xml reader for parametre.xml.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <libxml/parser.h>
  
```

```
#include <libxml/xpath.h>
#include "xml_parameter.h"

Include dependency graph for xml_parameter.c:
```



## Functions

- `xmlConfig_t * Xml_loadConfig (char *fichier)`  
*Initialize and load the parameter.xml.*
- `void Xml_freeConfig (xmlConfig_t *conf)`  
*Free the Struct `xmlConfig_t`.*
- `int Xml_getNumber (xmlConfig_t *conf, const char *requete)`  
*Get integer value.*
- `double Xml_getDoubleNumber (xmlConfig_t *conf, const char *requete)`  
*Get double value.*
- `char * XmlGetString (xmlConfig_t *conf)`  
*Get name value.*
- `int Xml_getNbSimulations (xmlConfig_t *conf)`  
*Get number of simulation.*
- `int Xml_getNbTriesMod (xmlConfig_t *conf)`  
*Get number of tries for modelling.*
- `int Xml_getNbTriesSa (xmlConfig_t *conf)`  
*Get number of tries for the simulated annealing and minimization.*
- `int Xml_getNbIter (xmlConfig_t *conf)`  
*Get number of iteration.*
- `int Xml_getNbGroup (xmlConfig_t *conf)`  
*Get number of group.*
- `double Xml_getStepSize (xmlConfig_t *conf)`  
*Get the step size.*
- `double Xml_getBoltzmann (xmlConfig_t *conf)`  
*Get the Boltzmann value.*
- `double Xml_getTinitial (xmlConfig_t *conf)`  
*Get the initial temperature.*
- `double Xml_getMuT (xmlConfig_t *conf)`

- `double Xml_getTmin (xmlConfig_t *conf)`  
*Get the variation of temperature.*
- `int Xml_getCount (xmlConfig_t *conf, const char *requete)`  
*Get the minimum temperature.*
- `int Xml_getNbCouples (xmlConfig_t *conf)`  
*Get count.*
- `int Xml_getNbParameters (xmlConfig_t *conf)`  
*Get the number of couples.*
- `int Xml_getNbReactioninNoeud (xmlConfig_t *conf, int noeud)`  
*Get the number of parameters.*
- `int Xml_getNbReaction (xmlConfig_t *conf)`  
*Count the number of reaction in one node.*
- `int Xml_getNbSpecies (xmlConfig_t *conf)`  
*Get the number of reactions.*
- `int Xml_getNbEquations (xmlConfig_t *conf)`  
*Get the number of species.*
- `int Xml_getNbBannedSpecies (xmlConfig_t *conf)`  
*Get the number of equations.*
- `char ** Xml_getReactionsNamesinNoeud (xmlConfig_t *conf, int numero)`  
*Get the number of banned species.*
- `char ** Xml_getReactionsNames (xmlConfig_t *conf)`  
*Get list of reactions.*
- `char ** Xml_getBanned (xmlConfig_t *conf)`  
*Get name of reactions.*
- `char ** Xml_getSpeciesFinalAmount (xmlConfig_t *conf, char *species)`  
*Get list of banned species.*
- `int * Xml_getallSpeciesFinalAmount (xmlConfig_t *conf, char **species, int taille)`  
*Get the final amount of one species.*
  
- `int Xml_getSpeciesWeight (xmlConfig_t *conf, char *species)`  
*Get Table of species amount.*
- `int * Xml_getallSpeciesWeight (xmlConfig_t *conf, char **species, int taille)`  
*Get Species weight.*
- `char ** Xml_allocEquation (char **equation, int nb_noeud)`  
*Get list of species weight.*
- `int Xml_getKineticLawNodes (xmlConfig_t *conf, int num_equation)`  
*Allocate memory for the list of equation.*
- `char *** Xml_getKineticLaw (xmlConfig_t *conf, int num_equation, int nb_noeud)`  
*Get the kinetic law for one node.*
  
- `Get list of kinetic laws.`

### 4.21.1 Detailed Description

Xml reader for parametre.xml. This file is part of MetaBoFlux (<http://www.cbib.u-bordeaux2.fr/metaboflux/>)  
Copyright (C) 2010 Amine Ghazlane from LaBRI and University of Bordeaux 1

MetaBoFlux is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MetaBoFlux is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

{Amine Ghazlane}

#### Version

2.0

#### Date

15 janvier 2010

Definition in file [xml\\_parameter.c](#).

### 4.21.2 Function Documentation

#### 4.21.2.1 `char ** Xml_allocEquation ( char ** equation, int nb_noeud )`

Allocate memory for the list of equation.

#### Author

Amine Ghazlane

#### Parameters

<code>equation</code>	list of equation
<code>nb_noeud</code>	Number of the node

#### Returns

List of reactions

Definition at line 789 of file [xml\\_parameter.c](#).

Referenced by `Xml_getKineticLaw()`.

#### 4.21.2.2 void Xml\_freeConfig ( *xmlConfig\_t* \* *conf* )

Free the Struct [xmlConfig\\_t](#).

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

Definition at line 97 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`, `xmlConfig_t::doc`, and `xmlConfig_t::fichier`.

Referenced by `Data_desallocation()`, and `Xml_loadConfig()`.

#### 4.21.2.3 int \* Xml\_getallSpeciesFinalAmount ( *xmlConfig\_t* \* *conf*, *char* \*\* *species*, *int* *taille* )

Get Table of species amount.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>species</i>	Table of species
<i>taille</i>	Number of species

##### Returns

Table of species amount

Definition at line 708 of file `xml_parameter.c`.

References `Xml_getSpeciesFinalAmount()`.

Referenced by `Data_allocScore()`.

Here is the call graph for this function:



**4.21.2.4 int \* Xml\_getallSpeciesWeight ( xmlConfig\_t \* conf, char \*\* species, int taille )**

Get list of species weight.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>species</i>	Table of species
<i>taille</i>	Number of species

**Returns**

List of species weight

Definition at line 770 of file [xml\\_parameter.c](#).

References [Xml\\_getSpeciesWeight\(\)](#).

Referenced by [Data\\_allocScore\(\)](#).

Here is the call graph for this function:

**4.21.2.5 char \*\* Xml\_getBanned ( xmlConfig\_t \* conf )**

Get list of banned species.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

List of banned species

Definition at line 624 of file xml\_parameter.c.

References `xmlConfig_t::ctxt`, and `Xml_getNbBannedSpecies()`.

Referenced by `Data_allocParameters()`.

Here is the call graph for this function:



#### 4.21.2.6 double `Xml_getBoltzmann ( xmlConfig_t * conf )`

Get the Boltzmann value.

##### Author

Amine Ghozlane

##### Parameters

<code>conf</code>	Struct <a href="#">xmlConfig_t</a>
-------------------	------------------------------------

##### Returns

Boltzmann value

Definition at line 305 of file xml\_parameter.c.

References `Xml_getDoubleNumber()`.

Referenced by `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:



**4.21.2.7 int Xml\_getCount ( *xmlConfig\_t* \* *conf*, const char \* *requete* )**

Get count.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>requete</i>	Xpath query

**Returns**

Count

Definition at line 358 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Xml_getNbBannedSpecies()`, `Xml_getNbCouples()`, `Xml_getNbEquations()`, `Xml_getNbParameters()`, `Xml_getNbReaction()`, `Xml_getNbReactioninNoeud()`, and `Xml_getNbSpecies()`.

**4.21.2.8 double Xml\_getDoubleNumber ( *xmlConfig\_t* \* *conf*, const char \* *requete* )**

Get double value.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>requete</i>	Xpath query

**Returns**

Read double value in the xml

Definition at line 153 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Xml_getBoltzmann()`, `Xml_getMuT()`, `Xml_getStepSize()`, `Xml_getTinitial()`, and `Xml_getTmin()`.

**4.21.2.9 char \*\*\* Xml\_getKineticLaw ( *xmlConfig\_t* \* *conf*, int *num\_equation*, int *nb\_noeud* )**

Get list of kinetic laws.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>num_-equation</i>	Number of equation
<i>nb_noeud</i>	Number nodes

**Returns**

List of kinetic laws

Definition at line 854 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`, and `Xml_allocEquation()`.

Referenced by `Data_allocEquations()`.

Here is the call graph for this function:



#### 4.21.2.10 int Xml\_getKineticLawNodes ( `xmlConfig_t * conf, int num_equation` )

Get the kinetic law for one node.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>num_-equation</i>	Number of the node

**Returns**

Ninetic law for one node

Definition at line 813 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Data_allocEquations()`.

#### 4.21.2.11 double `Xml_getMuT ( xmlConfig_t * conf )`

Get the variation of temperature.

##### Author

Amine Ghozlane

##### Parameters

<code>conf</code>	Struct <a href="#">xmlConfig_t</a>
-------------------	------------------------------------

##### Returns

Variation of temperature

Definition at line 331 of file `xml_parameter.c`.

References `Xml_getDoubleNumber()`.

Referenced by `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:



#### 4.21.2.12 int `Xml_getNbBannedSpecies ( xmlConfig_t * conf )`

Get the number of banned species.

##### Author

Amine Ghozlane

##### Parameters

<code>conf</code>	Struct <a href="#">xmlConfig_t</a>
-------------------	------------------------------------

**Returns**

Number of banned species

Definition at line 486 of file xml\_parameter.c.

References Xml\_getCount().

Referenced by Data\_allocParameters(), and Xml\_getBanned().

Here is the call graph for this function:

**4.21.2.13 int Xml\_getNbCouples ( xmlConfig\_t \* conf )**

Get the number of couples.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

Number of couples

Definition at line 396 of file xml\_parameter.c.

References Xml\_getCount().

Referenced by Data\_allocParameters(), and Xml\_getReactionsNames().

Here is the call graph for this function:



#### 4.21.2.14 int Xml\_getNbEquations ( xmlConfig\_t \* conf )

Get the number of equations.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

##### Returns

Number of equations

Definition at line 473 of file `xml_parameter.c`.

References `Xml_getCount()`.

Referenced by `Data_allocEquations()`.

Here is the call graph for this function:



#### 4.21.2.15 int Xml\_getNbGroup ( xmlConfig\_t \* conf )

Get number of group.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

Number of iteration

Definition at line 279 of file `xml_parameter.c`.

References `Xml_getNumber()`.

Referenced by `Mpi_master()`.

Here is the call graph for this function:

**4.21.2.16 int Xml\_getNbIter ( `xmlConfig_t` \* *conf* )**

Get number of iteration.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

Number of iteration

Definition at line 266 of file `xml_parameter.c`.

References `Xml_getNumber()`.

Referenced by `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:



#### 4.21.2.17 int Xml\_getNbParameters ( *xmlConfig\_t* \* *conf* )

Get the number of parameters.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

##### Returns

Number of parameters

Definition at line 409 of file `xml_parameter.c`.

References `Xml_getCount()`.

Referenced by `Data_allocParameters()`.

Here is the call graph for this function:



#### 4.21.2.18 int Xml\_getNbReaction ( *xmlConfig\_t* \* *conf* )

Get the number of reactions.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

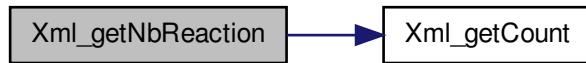
Number of reactions

Definition at line 447 of file `xml_parameter.c`.

References `Xml_getCount()`.

Referenced by `Data_allocScore()`, and `Xml_getReactionsNames()`.

Here is the call graph for this function:



#### 4.21.2.19 int Xml\_getNbReactioninNoeud ( `xmlConfig_t * conf`, `int noeud` )

Count the number of reaction in one node.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>noeud</i>	Number of the node

**Returns**

Number of reaction in one node

Definition at line 423 of file `xml_parameter.c`.

References `Xml_getCount()`.

Referenced by `Data_allocParameters()`, and `Xml_getReactionsNamesinNoeud()`.

Here is the call graph for this function:



#### 4.21.2.20 int Xml\_getNbSimulations ( *xmlConfig\_t* \* *conf* )

Get number of simulation.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

##### Returns

Number of simulation

Definition at line 227 of file [xml\\_parameter.c](#).

References [Xml\\_getNumber\(\)](#).

Referenced by [Mpi\\_master\(\)](#).

Here is the call graph for this function:



#### 4.21.2.21 int Xml\_getNbSpecies ( *xmlConfig\_t* \* *conf* )

Get the number of species.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

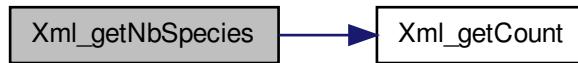
Number of species

Definition at line 460 of file `xml_parameter.c`.

References `Xml_getCount()`.

Referenced by `Data_allocScore()`, `Mpi_slave()`, and `Mpi_writer()`.

Here is the call graph for this function:

**4.21.2.22 int Xml\_getNbTriesMod ( xmlConfig\_t \* *conf* )**

Get number of tries for modelling.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

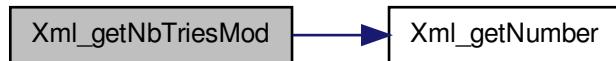
Number of tries

Definition at line 240 of file `xml_parameter.c`.

References `Xml_getNumber()`.

Referenced by `Data_allocParameters()`, `Mod_compute_modeling()`, `Mpi_sizeResultTab()`, and `Sd_compute_standard_deviation()`.

Here is the call graph for this function:



#### 4.21.2.23 int Xml\_getNbTriesSa ( *xmlConfig\_t* \* *conf* )

Get number of tries for the simulated annealing and minimization.

##### Author

Amine Ghozlane

##### Parameters

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

##### Returns

Number of tries

Definition at line 253 of file `xml_parameter.c`.

References `Xml_getNumber()`.

Referenced by `Data_allocParameters()`, `Min_my_f()`, `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:



#### 4.21.2.24 int Xml\_getNumber ( *xmlConfig\_t* \* *conf*, const char \* *requete* )

Get integer value.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>requete</i>	Xpath query

**Returns**

Read integer value in the xml

Definition at line 115 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Xml_getNbGroup()`, `Xml_getNbIter()`, `Xml_getNbSimulations()`, `Xml_getNbTriesMod()`, and `Xml_getNbTriesSa()`.

#### 4.21.2.25 `char ** Xml_getReactionsNames ( xmlConfig_t * conf )`

Get name of reactions.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

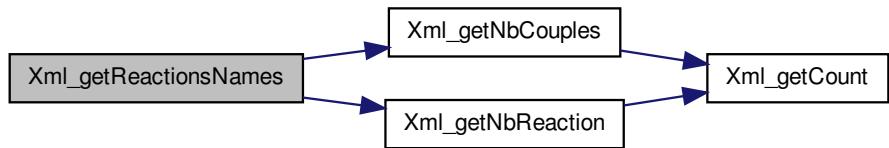
Name of reactions

Definition at line 541 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`, `Xml_getNbCouples()`, and `Xml_getNbReaction()`.

Referenced by `Data_allocScore()`.

Here is the call graph for this function:



#### 4.21.2.26 `char ** Xml_getReactionsNamesinNoeud ( xmlConfig_t * conf, int numero )`

Get list of reactions.

Get list of species.

#### Author

Amine Ghozlane

#### Parameters

<code>conf</code>	Struct <a href="#">xmlConfig_t</a>
<code>numero</code>	Number of the node

#### Returns

List of reactions

#### Author

Amine Ghozlane

#### Parameters

<code>conf</code>	Struct <a href="#">xmlConfig_t</a>
-------------------	------------------------------------

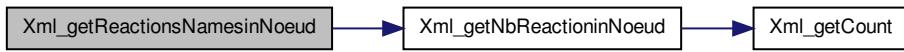
#### Returns

List of species

Definition at line 500 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`, and `Xml_getNbReactioninNoeud()`.

Here is the call graph for this function:



#### 4.21.2.27 `int Xml_getSpeciesFinalAmount ( xmlConfig_t * conf, char * species )`

Get the final amount of one species.

#### Author

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>species</i>	Name of one species

**Returns**

Final amount of one species

Definition at line 665 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Xml_getallSpeciesFinalAmount()`.

#### 4.21.2.28 int Xml\_getSpeciesWeight ( `xmlConfig_t * conf, char * species` )

Get Species weight.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
<i>species</i>	Name of one species

**Returns**

Species weight

Definition at line 727 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`.

Referenced by `Xml_getallSpeciesWeight()`.

#### 4.21.2.29 double Xml\_getStepSize ( `xmlConfig_t * conf` )

Get the step size.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

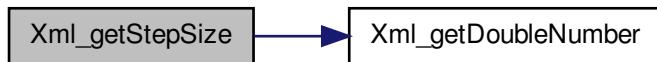
Step size

Definition at line 292 of file xml\_parameter.c.

References `Xml_getDoubleNumber()`.

Referenced by `Mpi_slave()`, and `Recuit_printParametre()`.

Here is the call graph for this function:



#### 4.21.2.30 `char * XmlGetString ( xmlConfig_t * conf )`

Get name value.

##### Author

Amine Ghozlane

##### Parameters

<code>conf</code>	Struct <code>xmlConfig_t</code>
-------------------	---------------------------------

##### Returns

Read name value in the xml

Definition at line 190 of file xml\_parameter.c.

References `xmlConfig_t::ctxt`.

Referenced by `Recuit_redirectionFlux()`.

#### 4.21.2.31 `double Xml_getTinitial ( xmlConfig_t * conf )`

Get the initial temperature.

##### Author

Amine Ghozlane

##### Parameters

<code>conf</code>	Struct <code>xmlConfig_t</code>
-------------------	---------------------------------

**Returns**

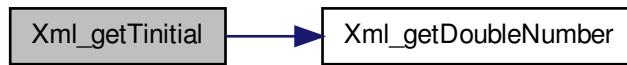
Initial temperature

Definition at line 318 of file xml\_parameter.c.

References Xml\_getDoubleNumber().

Referenced by Mpi\_slave(), and Recuit\_printParametre().

Here is the call graph for this function:

**4.21.2.32 double Xml\_getTmin ( xmlConfig\_t \* conf )**

Get the minimum temperature.

**Author**

Amine Ghozlane

**Parameters**

<i>conf</i>	Struct <a href="#">xmlConfig_t</a>
-------------	------------------------------------

**Returns**

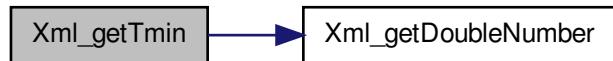
Minimum temperature

Definition at line 344 of file xml\_parameter.c.

References Xml\_getDoubleNumber().

Referenced by Mpi\_slave(), and Recuit\_printParametre().

Here is the call graph for this function:



#### 4.21.2.33 `xmlConfig_t * Xml_loadConfig ( char * fichier )`

Initialize and load the parameter.xml.

##### Author

Amine Ghozlane

##### Parameters

<code>fichier</code>	Xml file name
----------------------	---------------

##### Returns

Struct [xmlConfig\\_t](#)

Definition at line 42 of file `xml_parameter.c`.

References `xmlConfig_t::ctxt`, `xmlConfig_t::doc`, `xmlConfig_t::fichier`, `xmlConfig_t::racine`, and `Xml_freeConfig()`.

Referenced by `Data_initXML()`.

Here is the call graph for this function:



# Index

activity  
    ListArguments, 9

add  
    equations.c, 151  
    equations.h, 40

addition  
    equations.h, 39

alloc\_arguments  
    MetaBoFlux.c, 202

argument\_analysis  
    MetaBoFlux.c, 202

banned  
    ListParameters, 11

check\_arguments  
    MetaBoFlux.c, 202

check\_common  
    MetaBoFlux.c, 203

compute\_mpi  
    mpi\_load.c, 209  
    mpi\_load.h, 87

conf  
    ListParameters, 11

constant  
    equations.h, 39

ctxt  
    xmlConfig\_t, 21

data  
    Equedata, 6

Data\_allocEquations  
    data\_parameters.c, 137  
    data\_parameters.h, 25

Data\_allocParameters  
    data\_parameters.c, 137  
    data\_parameters.h, 25

Data\_allocScore  
    data\_parameters.c, 138  
    data\_parameters.h, 26

Data\_allocSimParameters  
    data\_parameters.c, 139  
    data\_parameters.h, 27

Data\_copieTab  
    data\_parameters.c, 139  
    data\_parameters.h, 27

Data\_desallocation  
    data\_parameters.c, 140  
    data\_parameters.h, 28

Data\_desallocSim  
    data\_parameters.c, 140  
    data\_parameters.h, 29

Data\_equationsAlloc  
    data\_parameters.c, 141  
    data\_parameters.h, 29

Data\_equationsInit  
    data\_parameters.c, 141  
    data\_parameters.h, 30

Data\_initSBML  
    data\_parameters.c, 143  
    data\_parameters.h, 31

Data\_initXML  
    data\_parameters.c, 144  
    data\_parameters.h, 32

Data\_parameters  
    data\_parameters.c, 144  
    data\_parameters.h, 32

data\_parameters.c  
    Data\_allocEquations, 137  
    Data\_allocParameters, 137  
    Data\_allocScore, 138  
    Data\_allocSimParameters, 139  
    Data\_copieTab, 139  
    Data\_desallocation, 140  
    Data\_desallocSim, 140  
    Data\_equationsAlloc, 141  
    Data\_equationsInit, 141  
    Data\_initSBML, 143  
    Data\_initXML, 144  
    Data\_parameters, 144  
    Data\_rngAlloc, 145  
    Data\_scoreAlloc, 146

Data\_scoreFree, 146  
Data\_scoreInit, 147  
Data\_simParameters, 147  
Data\_updateTab, 148  
data\_parameters.h  
    Data\_allocEquations, 25  
    Data\_allocParameters, 25  
    Data\_allocScore, 26  
    Data\_allocSimParameters, 27  
    Data\_copieTab, 27  
    Data\_desallocation, 28  
    Data\_desallocSim, 29  
    Data\_equationsAlloc, 29  
    Data\_equationsInit, 30  
    Data\_initSBML, 31  
    Data\_initXML, 32  
    Data\_parameters, 32  
    Data\_rngAlloc, 33  
    Data\_scoreAlloc, 34  
    Data\_scoreFree, 34  
    Data\_scoreInit, 35  
    Data\_simParameters, 35  
    Data\_updateTab, 36  
Data\_rngAlloc  
    data\_parameters.c, 145  
    data\_parameters.h, 33  
Data\_scoreAlloc  
    data\_parameters.c, 146  
    data\_parameters.h, 34  
Data\_scoreFree  
    data\_parameters.c, 146  
    data\_parameters.h, 34  
Data\_scoreInit  
    data\_parameters.c, 147  
    data\_parameters.h, 35  
Data\_simParameters  
    data\_parameters.c, 147  
    data\_parameters.h, 35  
Data\_updateTab  
    data\_parameters.c, 148  
    data\_parameters.h, 36  
debug  
    ListArguments, 9  
debugFile  
    SimParameters, 18  
divide  
    equations.c, 152  
    equations.h, 40  
division  
    equations.h, 39  
doc  
    xmlConfig\_t, 21  
equal  
    equations.h, 39  
equality  
    equations.c, 152  
    equations.h, 40  
equation  
    ListParameters, 11  
Equations, 5  
    info, 6  
    type, 6  
equations.c  
    add, 151  
    divide, 152  
    equality, 152  
    Equations\_addOp, 152  
    Equations\_alloc, 153  
    Equations\_calcul, 154  
    Equations\_defiler, 155  
    Equations\_defilerSuiv, 155  
    Equations\_define, 156  
    Equations\_depiler, 157  
    Equations\_empiler, 158  
    Equations\_emptyOp, 158  
    Equations\_extractData, 159  
    Equations\_finalQuantite, 159  
    Equations\_findSpecies, 160  
    Equations\_pileFormation, 161  
    Equations\_print, 163  
    Equations\_priorite, 163  
    Equations\_resultat, 163  
    Equations\_sommet, 164  
    Equations\_vide, 164  
    inf, 165  
    inf\_equal, 165  
    multiply, 166  
    subtract, 166  
    sup, 166  
    sup\_equal, 167  
equations.h  
    add, 40  
    addition, 39  
    constant, 39  
    divide, 40  
    division, 39  
    equal, 39  
    equality, 40  
    Equations\_addOp, 41

Equations\_alloc, 42  
 Equations\_calcul, 42  
 Equations\_defiler, 43  
 Equations\_defilerSuiv, 44  
 Equations\_define, 44  
 Equations\_depiler, 45  
 Equations\_empiler, 46  
 Equations\_emptyOp, 46  
 Equations\_finalQuantite, 47  
 Equations\_findSpecies, 48  
 Equations\_pileFormation, 48  
 Equations\_print, 51  
 Equations\_priorite, 51  
 Equations\_resultat, 51  
 Equations\_sommet, 52  
 Equations\_vide, 52  
 Equtype, 39  
 inf, 53  
 inf\_equal, 53  
 inferior, 39  
 inferior\_equal, 39  
 multiplication, 39  
 multiply, 54  
 soustraction, 39  
 subtract, 54  
 sup, 54  
 sup\_equal, 55  
 superior, 39  
 superior\_equal, 39  
 variable, 39  
 Equations\_addOp  
     equations.c, 152  
     equations.h, 41  
 Equations\_alloc  
     equations.c, 153  
     equations.h, 42  
 Equations\_calcul  
     equations.c, 154  
     equations.h, 42  
 Equations\_defiler  
     equations.c, 155  
     equations.h, 43  
 Equations\_defilerSuiv  
     equations.c, 155  
     equations.h, 44  
 Equations\_define  
     equations.c, 156  
     equations.h, 44  
 Equations\_depiler  
     equations.c, 157  
                 equations.h, 45  
                 Equations\_empiler  
                     equations.c, 158  
                     equations.h, 46  
                 Equations\_emptyOp  
                     equations.c, 158  
                     equations.h, 46  
                 Equations\_extractData  
                     equations.c, 159  
                 Equations\_finalQuantite  
                     equations.c, 159  
                     equations.h, 47  
                 Equations\_findSpecies  
                     equations.c, 160  
                     equations.h, 48  
                 Equations\_pileFormation  
                     equations.c, 161  
                     equations.h, 48  
                 Equations\_print  
                     equations.c, 163  
                     equations.h, 51  
                 Equations\_priorite  
                     equations.c, 163  
                     equations.h, 51  
                 Equations\_resultat  
                     equations.c, 163  
                     equations.h, 51  
                 Equations\_sommet  
                     equations.c, 164  
                     equations.h, 52  
                 Equations\_vide  
                     equations.c, 164  
                     equations.h, 52  
                 Equdata, 6  
                     data, 6  
                     var, 7  
                 Equtype  
                     equations.h, 39  
                 error\_activity  
                     MetaBoFlux.c, 204  
                 Especies, 7  
                     id, 8  
                     quantite, 8  
                     system, 8  
                 especies.c  
                     Especies\_alloc, 169  
                     Especies\_allocReactions, 169  
                     Especies\_find, 170  
                     Especies\_free, 170  
                     Especies\_freeReactions, 171

Especies\_getNbReactions, 171  
Especies\_getQuantite, 172  
Especies\_print, 172  
Especies\_print\_2, 172  
Especies\_save, 173  
Especies\_scoreSpecies, 173  
Especies\_setQuantite, 173  
  
especies.h  
    Especies\_alloc, 57  
    Especies\_allocReactions, 57  
    Especies\_find, 58  
    Especies\_free, 58  
    Especies\_freeReactions, 59  
    Especies\_getNbReactions, 59  
    Especies\_getQuantite, 60  
    Especies\_print, 60  
    Especies\_print\_2, 60  
    Especies\_save, 61  
    Especies\_scoreSpecies, 61  
    Especies\_setQuantite, 61  
  
Especies\_alloc  
    especies.c, 169  
    especies.h, 57  
  
Especies\_allocReactions  
    especies.c, 169  
    especies.h, 57  
  
Especies\_find  
    especies.c, 170  
    especies.h, 58  
  
Especies\_free  
    especies.c, 170  
    especies.h, 58  
  
Especies\_freeReactions  
    especies.c, 171  
    especies.h, 59  
  
Especies\_getNbReactions  
    especies.c, 171  
    especies.h, 59  
  
Especies\_getQuantite  
    especies.c, 172  
    especies.h, 60  
  
Especies\_print  
    especies.c, 172  
    especies.h, 60  
  
Especies\_print\_2  
    especies.c, 172  
    especies.h, 60  
  
Especies\_save  
    especies.c, 173  
    especies.h, 61  
  
Especies\_scoreSpecies  
    especies.c, 173  
    especies.h, 61  
  
Especies\_setQuantite  
    especies.c, 173  
    especies.h, 61  
  
fichier  
    xmlConfig\_t, 21  
  
files\_path  
    ListArguments, 9  
  
free\_arguments  
    MetaBoFlux.c, 204  
  
group  
    ListArguments, 9  
  
gsl\_min.c  
    Min\_compute\_minimization, 176  
    Min\_copieTab2, 177  
    Min\_copieTab3, 178  
    Min\_getTampon, 178  
    Min\_my\_f, 178  
    Min\_score\_print\_mean, 180  
    Min\_verifValue, 181  
  
gsl\_min.h  
    Min\_compute\_minimization, 63  
    Min\_copieTab2, 64  
    Min\_copieTab3, 65  
    Min\_getTampon, 65  
    Min\_my\_f, 66  
    Min\_score\_print\_mean, 67  
    Min\_verifValue, 68  
  
gsl\_mod.c  
    Mod\_compute\_modeling, 183  
    Mod\_score\_print\_mean, 184  
  
gsl\_mod.h  
    Mod\_compute\_modeling, 70  
    Mod\_score\_print\_mean, 71  
  
gsl\_recuit.c  
    Recuit\_compute\_recuit, 187  
    Recuit\_defParametre, 188  
    Recuit\_energyFunction, 189  
    Recuit\_metricDistance, 190  
    Recuit\_printParametre, 191  
    Recuit\_printPosition, 192  
    Recuit\_redirectionFlux, 193  
    Recuit\_takeStep, 193  
    Recuit\_verifParameters, 194  
    Recuit\_verifParameters\_2, 194  
  
gsl\_recuit.h

Recuit\_compute\_recuit, 74  
 Recuit\_defParametre, 75  
 Recuit\_energyFunction, 76  
 Recuit\_metricDistance, 77  
 Recuit\_printParametre, 78  
 Recuit\_printPosition, 79  
 Recuit\_redirectionFlux, 80  
 Recuit\_takeStep, 80  
 Recuit\_verifParameters, 81  
 Recuit\_verifParameters\_2, 81  
 gsl\_sd.c  
     Sd\_compute\_simulation, 197  
     Sd\_compute\_standard\_deviation, 198  
 gsl\_sd.h  
     Sd\_compute\_simulation, 83  
     Sd\_compute\_standard\_deviation, 84  
 help\_print  
     MetaBoFlux.c, 205  
 id  
     Especies, 8  
 inf  
     equations.c, 165  
     equations.h, 53  
 inf\_equal  
     equations.c, 165  
     equations.h, 53  
 inferior  
     equations.h, 39  
 inferior\_equal  
     equations.h, 39  
 info  
     Equations, 6  
 interest\_parameters  
     ListParameters, 11  
 link  
     Reaction, 14  
 ListArguments, 8  
     activity, 9  
     debug, 9  
     files\_path, 9  
     group, 9  
     port, 9  
 ListParameters, 10  
     banned, 11  
     conf, 11  
     equation, 11  
     interest\_parameters, 11  
 model, 11  
 nb\_banned, 12  
 nb\_couples, 12  
 nb\_equations, 12  
 nb\_parameters, 12  
 nb\_reaction, 12  
 nb\_triesMod, 12  
 nb\_triesSa, 13  
 noeud, 13  
 parameters, 13  
 main  
     MetaBoFlux.c, 205  
 MetaBoFlux.c  
     alloc\_arguments, 202  
     argument\_analysis, 202  
     check\_arguments, 202  
     check\_common, 203  
     error\_activity, 204  
     free\_arguments, 204  
     help\_print, 205  
     main, 205  
 metaboflux/include/data\_parameters.h, 23  
 metaboflux/include/equations.h, 37  
 metaboflux/include/especies.h, 55  
 metaboflux/include/gsl\_min.h, 62  
 metaboflux/include/gsl\_mod.h, 68  
 metaboflux/include/gsl\_recuit.h, 72  
 metaboflux/include/gsl\_sd.h, 82  
 metaboflux/include/mpi\_load.h, 86  
 metaboflux/include/simulation.h, 96  
 metaboflux/include/xml\_parameter.h, 111  
 metaboflux/src/data\_parameters.c, 134  
 metaboflux/src/equations.c, 149  
 metaboflux/src/especies.c, 167  
 metaboflux/src/gsl\_min.c, 174  
 metaboflux/src/gsl\_mod.c, 181  
 metaboflux/src/gsl\_recuit.c, 185  
 metaboflux/src/gsl\_sd.c, 195  
 metaboflux/src/MetaBoFlux.c, 199  
 metaboflux/src/mpi\_load.c, 206  
 metaboflux/src/simulation.c, 217  
 metaboflux/src/xml\_parameter.c, 232  
 Min\_compute\_minimization  
     gsl\_min.c, 176  
     gsl\_min.h, 63  
 Min\_copieTab2  
     gsl\_min.c, 177  
     gsl\_min.h, 64  
 Min\_copieTab3

gsl\_min.c, 178  
gsl\_min.h, 65  
Min\_getTampon  
    gsl\_min.c, 178  
    gsl\_min.h, 65  
Min\_my\_f  
    gsl\_min.c, 178  
    gsl\_min.h, 66  
Min\_score\_print\_mean  
    gsl\_min.c, 180  
    gsl\_min.h, 67  
Min\_verifValue  
    gsl\_min.c, 181  
    gsl\_min.h, 68  
minStepTab  
    TestReaction, 20  
Mod\_compute\_modeling  
    gsl\_mod.c, 183  
    gsl\_mod.h, 70  
Mod\_score\_print\_mean  
    gsl\_mod.c, 184  
    gsl\_mod.h, 71  
model  
    ListParameters, 11  
Mpi\_allocResultTab  
    mpi\_load.c, 210  
    mpi\_load.h, 89  
Mpi\_connectInterface  
    mpi\_load.c, 211  
    mpi\_load.h, 90  
Mpi\_disconnectInterface  
    mpi\_load.c, 211  
    mpi\_load.h, 90  
Mpi\_finalize  
    mpi\_load.c, 212  
    mpi\_load.h, 91  
Mpi\_init  
    mpi\_load.c, 212  
    mpi\_load.h, 91  
mpi\_load.c  
    compute\_mpi, 209  
    Mpi\_allocResultTab, 210  
    Mpi\_connectInterface, 211  
    Mpi\_disconnectInterface, 211  
    Mpi\_finalize, 212  
    Mpi\_init, 212  
    Mpi\_master, 212  
    Mpi\_sizeResultTab, 213  
    Mpi\_slave, 214  
    Mpi\_writer, 215  
Mpi\_writeSdFile, 216  
Mpi\_writeSimFile, 217  
mpi\_load.h  
    compute\_mpi, 87  
    Mpi\_allocResultTab, 89  
    Mpi\_connectInterface, 90  
    Mpi\_disconnectInterface, 90  
    Mpi\_finalize, 91  
    Mpi\_init, 91  
    Mpi\_master, 91  
    Mpi\_sizeResultTab, 92  
    Mpi\_slave, 93  
    Mpi\_writer, 94  
    Mpi\_writeSdFile, 95  
    Mpi\_writeSimFile, 96  
Mpi\_master  
    mpi\_load.c, 212  
    mpi\_load.h, 91  
Mpi\_sizeResultTab  
    mpi\_load.c, 213  
    mpi\_load.h, 92  
Mpi\_slave  
    mpi\_load.c, 214  
    mpi\_load.h, 93  
Mpi\_writer  
    mpi\_load.c, 215  
    mpi\_load.h, 94  
Mpi\_writeSdFile  
    mpi\_load.c, 216  
    mpi\_load.h, 95  
Mpi\_writeSimFile  
    mpi\_load.c, 217  
    mpi\_load.h, 96  
multiplication  
    equations.h, 39  
multiply  
    equations.c, 166  
    equations.h, 54  
name  
    Score, 15  
nb\_banned  
    ListParameters, 12  
nb\_couples  
    ListParameters, 12  
nb\_equations  
    ListParameters, 12  
nb\_parameters  
    ListParameters, 12  
nb\_reaction

ListParameters, 12  
 Score, 15  
 nb\_species  
     Score, 15  
 nb\_triesMod  
     ListParameters, 12  
 nb\_triesSa  
     ListParameters, 13  
 noeud  
     ListParameters, 13  
 option, 13  
 out  
     SimParameters, 18  
 parameters  
     ListParameters, 13  
 pile  
     SimParameters, 19  
 port  
     ListArguments, 9  
 quantite  
     Especies, 8  
     Score, 16  
 r  
     SimParameters, 19  
 racine  
     xmlConfig\_t, 21  
 Reaction, 14  
     link, 14  
     suivant, 14  
 reaction  
     Score, 16  
 Recuit\_compute\_recuit  
     gsl\_recuit.c, 187  
     gsl\_recuit.h, 74  
 Recuit\_defParametre  
     gsl\_recuit.c, 188  
     gsl\_recuit.h, 75  
 Recuit\_energyFunction  
     gsl\_recuit.c, 189  
     gsl\_recuit.h, 76  
 Recuit\_metricDistance  
     gsl\_recuit.c, 190  
     gsl\_recuit.h, 77  
 Recuit\_printParametre  
     gsl\_recuit.c, 191  
     gsl\_recuit.h, 78  
 Recuit\_printPosition  
     gsl\_recuit.c, 192  
     gsl\_recuit.h, 79  
 Recuit\_redirectionFlux  
     gsl\_recuit.c, 193  
     gsl\_recuit.h, 80  
 Recuit\_takeStep  
     gsl\_recuit.c, 193  
     gsl\_recuit.h, 80  
 Recuit\_verifParameters  
     gsl\_recuit.c, 194  
     gsl\_recuit.h, 81  
 Recuit\_verifParameters\_2  
     gsl\_recuit.c, 194  
     gsl\_recuit.h, 81  
 SBML\_allocTest  
     simulation.c, 220  
     simulation.h, 98  
 SBML\_checkQuantite  
     simulation.c, 220  
     simulation.h, 98  
 SBML\_compute\_simulation  
     simulation.c, 221  
     simulation.h, 99  
 SBML\_compute\_simulation\_mean  
     simulation.c, 222  
     simulation.h, 100  
 SBML\_debugPrint  
     simulation.c, 223  
     simulation.h, 102  
 SBML\_debugPrintHead  
     simulation.c, 224  
     simulation.h, 103  
 SBML\_EstimationReaction  
     simulation.c, 224  
     simulation.h, 103  
 SBML\_evalExpression  
     simulation.c, 225  
     simulation.h, 104  
 SBML\_findReaction  
     simulation.c, 225  
     simulation.h, 104  
 SBML\_freeTest  
     simulation.c, 226  
     simulation.h, 105  
 SBML\_initEspeceAmounts  
     simulation.c, 226  
     simulation.h, 105  
 SBML\_reactChoice

simulation.c, 227  
simulation.h, 106  
SBML\_reaction  
    simulation.c, 227  
    simulation.h, 106  
SBML\_score  
    simulation.c, 228  
    simulation.h, 107  
SBML\_score\_add  
    simulation.c, 229  
    simulation.h, 108  
SBML\_score\_mean  
    simulation.c, 230  
    simulation.h, 109  
SBML\_setReactions  
    simulation.c, 230  
    simulation.h, 109  
SBML\_simulate  
    simulation.c, 231  
    simulation.h, 110  
Score, 15  
    name, 15  
    nb\_reaction, 15  
    nb\_species, 15  
    quantite, 16  
    reaction, 16  
    species, 16  
    species\_amount, 16  
    species\_weight, 16  
    taille, 17  
    tailleReactions, 17  
    tailleSpecies, 17  
Sd\_compute\_simulation  
    gsl\_sd.c, 197  
    gsl\_sd.h, 83  
Sd\_compute\_standard\_deviation  
    gsl\_sd.c, 198  
    gsl\_sd.h, 84  
SimParameters, 17  
    debugFile, 18  
    out, 18  
    pile, 19  
    r, 19  
    y, 19  
simulation.c  
    SBML\_allocTest, 220  
    SBML\_checkQuantite, 220  
    SBML\_compute\_simulation, 221  
    SBML\_compute\_simulation\_mean, 222  
    SBML\_debugPrint, 223  
    SBML\_debugPrintHead, 224  
    SBML\_EstimationReaction, 224  
    SBML\_evalExpression, 225  
    SBML\_findReaction, 225  
    SBML\_freeTest, 226  
    SBML\_initEspeceAmounts, 226  
    SBML\_reactChoice, 227  
    SBML\_reaction, 227  
    SBML\_score, 228  
    SBML\_score\_add, 229  
    SBML\_score\_mean, 230  
    SBML\_setReactions, 230  
    SBML\_simulate, 231  
simulation.h  
    SBML\_allocTest, 98  
    SBML\_checkQuantite, 98  
    SBML\_compute\_simulation, 99  
    SBML\_compute\_simulation\_mean, 100  
    SBML\_debugPrint, 102  
    SBML\_debugPrintHead, 103  
    SBML\_EstimationReaction, 103  
    SBML\_evalExpression, 104  
    SBML\_findReaction, 104  
    SBML\_freeTest, 105  
    SBML\_initEspeceAmounts, 105  
    SBML\_reactChoice, 106  
    SBML\_reaction, 106  
    SBML\_score, 107  
    SBML\_score\_add, 108  
    SBML\_score\_mean, 109  
    SBML\_setReactions, 109  
    SBML\_simulate, 110  
soustraction  
    equations.h, 39  
species  
    Score, 16  
species\_amount  
    Score, 16  
species\_weight  
    Score, 16  
subtract  
    equations.c, 166  
    equations.h, 54  
suivant  
    Reaction, 14  
sup  
    equations.c, 166  
    equations.h, 54  
sup\_equal  
    equations.c, 167

equations.h, 55  
 superior  
     equations.h, 39  
 superior\_equal  
     equations.h, 39  
 system  
     Especies, 8  
  
 tabReactions  
     TestReaction, 20  
 taille  
     Score, 17  
 tailleReactions  
     Score, 17  
 tailleSpecies  
     Score, 17  
 TestReaction, 19  
     minStepTab, 20  
     tabReactions, 20  
 type  
     Equations, 6  
  
 var  
     Equdata, 7  
 variable  
     equations.h, 39  
  
 Xml\_allocEquation  
     xml\_parameter.c, 235  
     xml\_parameter.h, 114  
 Xml\_freeConfig  
     xml\_parameter.c, 235  
     xml\_parameter.h, 114  
 Xml\_getallSpeciesFinalAmount  
     xml\_parameter.c, 236  
     xml\_parameter.h, 115  
 Xml\_getallSpeciesWeight  
     xml\_parameter.c, 237  
     xml\_parameter.h, 115  
 Xml\_getBanned  
     xml\_parameter.c, 237  
     xml\_parameter.h, 116  
 Xml\_getBoltzmann  
     xml\_parameter.c, 238  
     xml\_parameter.h, 117  
 Xml\_getCount  
     xml\_parameter.c, 238  
     xml\_parameter.h, 117  
 Xml\_getDoubleNumber  
     xml\_parameter.c, 239  
  
     xml\_parameter.h, 118  
 Xml\_getKineticLaw  
     xml\_parameter.c, 239  
     xml\_parameter.h, 118  
 Xml\_getKineticLawNodes  
     xml\_parameter.c, 240  
     xml\_parameter.h, 119  
 Xml\_getMuT  
     xml\_parameter.c, 241  
     xml\_parameter.h, 119  
 Xml\_getNbBannedSpecies  
     xml\_parameter.c, 241  
     xml\_parameter.h, 120  
 Xml\_getNbCouples  
     xml\_parameter.c, 242  
     xml\_parameter.h, 121  
 Xml\_getNbEquations  
     xml\_parameter.c, 243  
     xml\_parameter.h, 121  
 Xml\_getNbGroup  
     xml\_parameter.c, 243  
     xml\_parameter.h, 122  
 Xml\_getNbIter  
     xml\_parameter.c, 244  
     xml\_parameter.h, 123  
 Xml\_getNbParameters  
     xml\_parameter.c, 245  
     xml\_parameter.h, 123  
 Xml\_getNbReaction  
     xml\_parameter.c, 245  
     xml\_parameter.h, 124  
 Xml\_getNbReactioninNoeud  
     xml\_parameter.c, 246  
     xml\_parameter.h, 125  
 Xml\_getNbSimulations  
     xml\_parameter.c, 247  
     xml\_parameter.h, 125  
 Xml\_getNbSpecies  
     xml\_parameter.c, 247  
     xml\_parameter.h, 126  
 Xml\_getNbTriesMod  
     xml\_parameter.c, 248  
     xml\_parameter.h, 127  
 Xml\_getNbTriesSa  
     xml\_parameter.c, 249  
     xml\_parameter.h, 127  
 Xml\_getNumber  
     xml\_parameter.c, 249  
     xml\_parameter.h, 128  
 Xml\_getReactionsNames

xml\_parameter.c, 250  
xml\_parameter.h, 129  
Xml\_getReactionsNamesinNoeud  
  xml\_parameter.c, 251  
  xml\_parameter.h, 129  
Xml\_getSpeciesFinalAmount  
  xml\_parameter.c, 251  
  xml\_parameter.h, 130  
Xml\_getSpeciesWeight  
  xml\_parameter.c, 252  
  xml\_parameter.h, 131  
Xml\_getStepSize  
  xml\_parameter.c, 252  
  xml\_parameter.h, 131  
Xml\_getString  
  xml\_parameter.c, 253  
  xml\_parameter.h, 132  
Xml\_getTinitial  
  xml\_parameter.c, 253  
  xml\_parameter.h, 132  
Xml\_getTmin  
  xml\_parameter.c, 254  
  xml\_parameter.h, 133  
Xml\_loadConfig  
  xml\_parameter.c, 255  
  xml\_parameter.h, 133  
xml\_parameter.c  
  Xml\_allocEquation, 235  
  Xml\_freeConfig, 235  
  Xml\_getallSpeciesFinalAmount, 236  
  Xml\_getallSpeciesWeight, 237  
  Xml\_getBanned, 237  
  Xml\_getBoltzmann, 238  
  Xml\_getCount, 238  
  Xml\_getDoubleNumber, 239  
  Xml\_getKineticLaw, 239  
  Xml\_getKineticLawNodes, 240  
  Xml\_getMuT, 241  
  Xml\_getNbBannedSpecies, 241  
  Xml\_getNbCouples, 242  
  Xml\_getNbEquations, 243  
  Xml\_getNbGroup, 243  
  Xml\_getNbIter, 244  
  Xml\_getNbParameters, 245  
  Xml\_getNbReaction, 245  
  Xml\_getNbReactioninNoeud, 246  
  Xml\_getNbSimulations, 247  
  Xml\_getNbSpecies, 247  
  Xml\_getNbTriesMod, 248  
  Xml\_getNbTriesSa, 249  
                Xml\_getNumber, 249  
                Xml\_getReactionsNames, 250  
                Xml\_getReactionsNamesinNoeud, 251  
                Xml\_getSpeciesFinalAmount, 251  
                Xml\_getSpeciesWeight, 252  
                Xml\_getStepSize, 252  
                Xml\_getString, 253  
                Xml\_getTinitial, 253  
                Xml\_getTmin, 254  
                Xml\_loadConfig, 255  
                xml\_parameter.h  
                    Xml\_allocEquation, 114  
                    Xml\_freeConfig, 114  
                    Xml\_getallSpeciesFinalAmount, 115  
                    Xml\_getallSpeciesWeight, 115  
                    Xml\_getBanned, 116  
                    Xml\_getBoltzmann, 117  
                    Xml\_getCount, 117  
                    Xml\_getDoubleNumber, 118  
                    Xml\_getKineticLaw, 118  
                    Xml\_getKineticLawNodes, 119  
                    Xml\_getMuT, 119  
                    Xml\_getNbBannedSpecies, 120  
                    Xml\_getNbCouples, 121  
                    Xml\_getNbEquations, 121  
                    Xml\_getNbGroup, 122  
                    Xml\_getNbIter, 123  
                    Xml\_getNbParameters, 123  
                    Xml\_getNbReaction, 124  
                    Xml\_getNbReactioninNoeud, 125  
                    Xml\_getNbSimulations, 125  
                    Xml\_getNbSpecies, 126  
                    Xml\_getNbTriesMod, 127  
                    Xml\_getNbTriesSa, 127  
                    Xml\_getNumber, 128  
                    Xml\_getReactionsNames, 129  
                    Xml\_getReactionsNamesinNoeud, 129  
                    Xml\_getSpeciesFinalAmount, 130  
                    Xml\_getSpeciesWeight, 131  
                    Xml\_getStepSize, 131  
                    Xml\_getString, 132  
                    Xml\_getTinitial, 132  
                    Xml\_getTmin, 133  
                    Xml\_loadConfig, 133  
                xmlConfig\_t, 20  
                    ctxt, 21  
                    doc, 21  
                    fichier, 21  
                    racine, 21

y

SimParameters, [19](#)