APEMoST

Generated by Doxygen 1.7.3

Sun Jan 8 2012 01:31:01

Contents

1 Main Page **1**
1.1 Compile-time parameters . . . 1
1.1.1 Fine-tuning the algorithm . . . 1
1.1.2 Defining algorithm behaviour . . . 2
1.1.3 Running . . . 2
1.1.4 Analyzing . . . 2
1.1.5 Others . . . 2
1.2 Runtime parameters . . . 2

2 Data Structure Index **3**
2.1 Data Structures . . . 3

3 File Index **5**
3.1 File List . . . 5

4 Data Structure Documentation **7**
4.1 mcmc Struct Reference . . . 7
4.1.1 Detailed Description . . . 7
4.1.2 Field Documentation . . . 8
4.1.2.1 accept . . . 8
4.1.2.2 additional_data . . . 8
4.1.2.3 data . . . 8
4.1.2.4 files . . . 8
4.1.2.5 n_iter . . . 8
4.1.2.6 n_par . . . 8
4.1.2.7 params . . . 9
4.1.2.8 params_accepts . . . 9
4.1.2.9 params_best . . . 9
4.1.2.10 params_descr . . . 9
4.1.2.11 params_max . . . 9
4.1.2.12 params_min . . . 9
4.1.2.13 params_rejects . . . 9
4.1.2.14 params_step . . . 10
4.1.2.15 prior . . . 10
4.1.2.16 prob . . . 10
4.1.2.17 prob_best . . . 10
4.1.2.18 random . . . 10
4.1.2.19 reject . . . 10
4.2 parallel_tempering_mcmc Struct Reference . . . 11

4.2.1 Field Documentation . . . 11
4.2.1.1 beta . . . 11
4.2.1.2 swapcount . . . 11
4.3 Problem Struct Reference . . . 11
4.3.1 Field Documentation . . . 11
4.3.1.1 LogLike . . . 11
4.3.1.2 Prior . . . 11

5 File Documentation **13**
5.1 apps/benchmark_main.c File Reference . . . 13
5.1.1 Function Documentation . . . 13
5.1.1.1 main . . . 13
5.2 apps/bernoulli_example.c File Reference . . . 13
5.2.1 Define Documentation . . . 14
5.2.1.1 SIGMA . . . 14
5.2.2 Function Documentation . . . 14
5.2.2.1 calc_model . . . 14
5.2.2.2 calc_model_for . . . 14
5.3 apps/eval_main.c File Reference . . . 15
5.3.1 Function Documentation . . . 15
5.3.1.1 main . . . 15
5.4 apps/generic_main.c File Reference . . . 15
5.4.1 Define Documentation . . . 16
5.4.1.1 MAX_ITERATIONS . . . 16
5.4.1.2 OUTPUT_PARAMD . . . 16
5.4.1.3 OUTPUT_PARAMI . . . 16
5.4.2 Function Documentation . . . 16
5.4.2.1 check . . . 16
5.4.2.2 checkfile . . . 17
5.4.2.3 help_phase . . . 17
5.4.2.4 main . . . 17
5.4.2.5 usage . . . 17
5.4.3 Variable Documentation . . . 17
5.4.3.1 progname . . . 17
5.5 apps/library.c File Reference . . . 17
5.5.1 Function Documentation . . . 18
5.5.1.1 calc_model . . . 18
5.5.1.2 calc_model_for . . . 18
5.5.1.3 set_function . . . 18
5.5.2 Variable Documentation . . . 18
5.5.2.1 p . . . 18
5.6 apps/normal.c File Reference . . . 19
5.6.1 Function Documentation . . . 19
5.6.1.1 calc_model . . . 19
5.6.1.2 calc_model_for . . . 19
5.7 apps/pulse.c File Reference . . . 19
5.7.1 Define Documentation . . . 20
5.7.1.1 HMIN . . . 20
5.7.2 Function Documentation . . . 20
5.7.2.1 calc_model . . . 20

5.7.2.2 calc_model_for 20
5.8 apps/pulse_vrot.c File Reference 21
5.8.1 Define Documentation 21
5.8.1.1 HMIN 21
5.8.2 Function Documentation 21
5.8.2.1 calc_model 21
5.8.2.2 calc_model_for 21
5.9 apps/simplesin.c File Reference 22
5.9.1 Define Documentation 22
5.9.1.1 SIGMA 22
5.9.2 Function Documentation 22
5.9.2.1 apply_formula 22
5.9.2.2 calc_model 22
5.9.2.3 calc_model_for 23
5.10 apps/simplesin2.c File Reference 23
5.10.1 Define Documentation 23
5.10.1.1 SIGMA 23
5.10.2 Function Documentation 24
5.10.2.1 apply_formula 24
5.10.2.2 calc_model 24
5.10.2.3 calc_model_for 24
5.11 apps/simplesin5.c File Reference 24
5.11.1 Define Documentation 25
5.11.1.1 SIGMA 25
5.11.2 Function Documentation 25
5.11.2.1 apply_formula 25
5.11.2.2 calc_model 25
5.11.2.3 calc_model_for 25
5.12 src/analyse.c File Reference 26
5.12.1 Define Documentation 26
5.12.1.1 GNUPLOT_STYLE 26
5.12.1.2 HISTOGRAMS_MINMAX 26
5.12.1.3 NBINS 27
5.12.2 Function Documentation 27
5.12.2.1 analyse_data_probability 27
5.12.2.2 analyse_marginal_distributions 27
5.12.2.3 calc_marginal_distribution 27
5.12.2.4 calc_mcmc_error 27
5.13 src/analyse.h File Reference 27
5.13.1 Function Documentation 28
5.13.1.1 analyse_data_probability 28
5.13.1.2 analyse_marginal_distributions 28
5.14 src/debug.c File Reference 28
5.14.1 Function Documentation 28
5.14.1.1 dump_mcmc 28
5.14.1.2 dump_vector 28
5.14.1.3 dump_vectorln 28
5.15 src/debug.h File Reference 28
5.15.1 Define Documentation 30
5.15.1.1 AT 30

5.15.1.2 DEBUG . . . 30
5.15.1.3 debug . . . 30
5.15.1.4 dump_d . . . 30
5.15.1.5 dump_i . . . 30
5.15.1.6 dump_i_s . . . 30
5.15.1.7 dump_l . . . 30
5.15.1.8 dump_m . . . 30
5.15.1.9 dump_p . . . 30
5.15.1.10 dump_s . . . 31
5.15.1.11 dump_size . . . 31
5.15.1.12 dump_ui . . . 31
5.15.1.13 dump_ul . . . 31
5.15.1.14 dump_v . . . 31
5.15.1.15 IFDEBUG . . . 31
5.15.1.16 IFSEGV . . . 31
5.15.1.17 IFVERBOSE . . . 31
5.15.1.18 IFWAIT . . . 31
5.15.1.19 require . . . 31
5.15.1.20 SEGV . . . 32
5.15.1.21 STRINGIFY . . . 32
5.15.1.22 TOSTRING . . . 32
5.15.1.23 VERBOSE . . . 32
5.15.2 Function Documentation . . . 32
5.15.2.1 dump_mcmc . . . 32
5.15.2.2 dump_vector . . . 32
5.15.2.3 dump_vectorln . . . 32
5.16 src/define_defaults.h File Reference . . . 33
5.16.1 Define Documentation . . . 33
5.16.1.1 BETA_0 . . . 33
5.16.1.2 DATA_FILENAME . . . 33
5.16.1.3 ITER_LIMIT . . . 33
5.16.1.4 MAX_AR_DEVIATION . . . 33
5.16.1.5 MUL . . . 33
5.16.1.6 N_BETA . . . 34
5.16.1.7 N_SWAP . . . 34
5.16.1.8 PARAMS_FILENAME . . . 34
5.16.1.9 TARGET_ACCEPTANCE_RATE . . . 34
5.17 src/gsl_helper.c File Reference . . . 34
5.17.1 Function Documentation . . . 35
5.17.1.1 calc_deviation . . . 35
5.17.1.2 calc_normalized . . . 35
5.17.1.3 calc_same . . . 35
5.17.1.4 calc_vector_squaresum . . . 35
5.17.1.5 calc_vector_sum . . . 35
5.17.1.6 dup_vector . . . 36
5.17.1.7 linreg_n . . . 36
5.17.1.8 max_column . . . 36
5.17.1.9 max_row . . . 36
5.17.1.10 max_vector . . . 36
5.17.1.11 min_column . . . 36

5.17.1.12 min_row 36
5.17.1.13 min_vector 37
5.17.1.14 sort 37
5.17.1.15 xbar 37
5.17.1.16 xbar_j 37
5.18 src/gsl_helper.h File Reference 37
5.18.1 Function Documentation 38
5.18.1.1 calc_deviation 38
5.18.1.2 calc_normalized 38
5.18.1.3 calc_same 38
5.18.1.4 calc_vector_squaresum 38
5.18.1.5 calc_vector_sum 38
5.18.1.6 dup_vector 39
5.18.1.7 linreg_n 39
5.18.1.8 max_column 39
5.18.1.9 max_row 39
5.18.1.10 max_vector 39
5.18.1.11 min_column 39
5.18.1.12 min_row 39
5.18.1.13 min_vector 40
5.18.1.14 sort 40
5.19 src/histogram.c File Reference 40
5.19.1 Function Documentation 41
5.19.1.1 append_to_hists 41
5.19.1.2 calc_hist 41
5.19.1.3 create_hist 41
5.19.1.4 find_min_max 41
5.19.1.5 update_min_max 41
5.20 src/histogram.h File Reference 42
5.20.1 Function Documentation 42
5.20.1.1 append_to_hists 42
5.20.1.2 calc_hist 42
5.20.1.3 create_hist 42
5.20.1.4 find_min_max 42
5.20.1.5 update_min_max 43
5.21 src/markov_chain.c File Reference 43
5.21.1 Define Documentation 44
5.21.1.1 MAXIMAL_STEPWIDTH 44
5.21.1.2 MINIMAL_STEPWIDTH 44
5.21.2 Function Documentation 44
5.21.2.1 assess_acceptance_rate 44
5.21.2.2 burn_in 45
5.21.2.3 clear_bit 45
5.21.2.4 do_step_for 45
5.21.2.5 get_bit 45
5.21.2.6 markov_chain_step 45
5.21.2.7 markov_chain_step_for 45
5.21.2.8 restart_from_best 46
5.21.2.9 rmw_adapt_stepwidth 46
5.21.2.10 set_bit 46

5.22 src/markov_chain.h File Reference 46
5.22.1 Define Documentation 47
5.22.1.1 ACCURACY_DEVIATION_FACTOR 47
5.22.1.2 CIRCULAR_PARAMS 47
5.22.1.3 DEFAULT_ADJUST_STEP 47
5.22.1.4 ITER_READJUST 47
5.22.1.5 NO_RESCALING_LIMIT 47
5.22.2 Function Documentation 48
5.22.2.1 assess_acceptance_rate 48
5.22.2.2 burn_in 49
5.22.2.3 markov_chain_calibrate 49
5.22.2.4 markov_chain_step 49
5.22.2.5 markov_chain_step_for 50
5.22.2.6 restart_from_best 50
5.22.2.7 rmw_adapt_stepwidth 50
5.23 src/markov_chain_calibrate.c File Reference 50
5.23.1 Define Documentation 51
5.23.1.1 BETWEEN 51
5.23.1.2 MAX 51
5.23.1.3 MAX_ACCURACY_IMPROVEMENT 51
5.23.1.4 MIN 52
5.23.1.5 SCALE_LIN_WORST 52
5.23.1.6 SCALE_MIN 52
5.23.2 Function Documentation 52
5.23.2.1 markov_chain_calibrate 52
5.23.2.2 markov_chain_calibrate_alt 52
5.23.2.3 markov_chain_calibrate_linear_regression 53
5.23.2.4 markov_chain_calibrate_multilinear_regression . . 53
5.23.2.5 markov_chain_calibrate_orig 53
5.23.2.6 markov_chain_calibrate_quadratic 53
5.24 src/mcmc.c File Reference 54
5.24.1 Function Documentation 54
5.24.1.1 init_seed 54
5.24.1.2 mcmc_check 54
5.24.1.3 mcmc_free 54
5.24.1.4 mcmc_init 55
5.24.2 Variable Documentation 55
5.24.2.1 r 55
5.25 src/mcmc.h File Reference 55
5.25.1 Define Documentation 56
5.25.1.1 assert 56
5.25.1.2 DUMP_FORMAT 56
5.25.1.3 NOASSERT 57
5.25.2 Function Documentation 57
5.25.2.1 calc_model 57
5.25.2.2 calc_model_for 57
5.25.2.3 mcmc_append_current_parameters 57
5.25.2.4 mcmc_check 58
5.25.2.5 mcmc_check_best 58
5.25.2.6 mcmc_dump_close 58

5.25.2.7 mcmc_dump_current . . . 58
5.25.2.8 mcmc_dump_flush . . . 58
5.25.2.9 mcmc_dump_probabilities . . . 59
5.25.2.10 mcmc_dump_y_dat . . . 59
5.25.2.11 mcmc_free . . . 59
5.25.2.12 mcmc_load . . . 59
5.25.2.13 mcmc_load_data . . . 60
5.25.2.14 mcmc_load_params . . . 60
5.25.2.15 mcmc_open_dump_files . . . 60
5.25.2.16 mcmc_reuse_data . . . 61
5.26 src/mcmc_calculate.c File Reference . . . 61
5.26.1 Function Documentation . . . 61
5.26.1.1 mcmc_append_current_parameters . . . 61
5.26.1.2 mcmc_check_best . . . 62
5.27 src/mcmc_dump.c File Reference . . . 62
5.27.1 Define Documentation . . . 62
5.27.1.1 ASSURE_DUMP_ENABLED . . . 62
5.27.2 Function Documentation . . . 63
5.27.2.1 mcmc_dump_close . . . 63
5.27.2.2 mcmc_dump_current . . . 63
5.27.2.3 mcmc_dump_flush . . . 63
5.27.2.4 mcmc_dump_y_dat . . . 63
5.27.2.5 mcmc_open_dump_files . . . 63
5.28 src/mcmc_gettersetter.c File Reference . . . 64
5.28.1 Define Documentation . . . 65
5.28.1.1 get_n_par . . . 65
5.28.1.2 N_PARAMETERS . . . 65
5.28.1.3 PROPOSAL . . . 66
5.28.2 Function Documentation . . . 66
5.28.2.1 free_gsl_vector_array . . . 66
5.28.2.2 get_accept_rate . . . 66
5.28.2.3 get_accept_rate_for . . . 66
5.28.2.4 get_accept_rate_global . . . 66
5.28.2.5 get_data . . . 66
5.28.2.6 get_next_alog_urandom . . . 66
5.28.2.7 get_next_random_jump . . . 66
5.28.2.8 get_next_uniform_plusminus_random . . . 67
5.28.2.9 get_next_uniform_random . . . 67
5.28.2.10 get_params . . . 67
5.28.2.11 get_params_accepts_for . . . 67
5.28.2.12 get_params_accepts_global . . . 67
5.28.2.13 get_params_accepts_sum . . . 67
5.28.2.14 get_params_best . . . 67
5.28.2.15 get_params_best_for . . . 68
5.28.2.16 get_params_descr . . . 68
5.28.2.17 get_params_for . . . 68
5.28.2.18 get_params_max . . . 68
5.28.2.19 get_params_max_for . . . 68
5.28.2.20 get_params_min . . . 68
5.28.2.21 get_params_min_for . . . 68

5.28.2.22 get_params_rejects_for 69
5.28.2.23 get_params_rejects_global 69
5.28.2.24 get_params_rejects_sum 69
5.28.2.25 get_prior 69
5.28.2.26 get_prob 69
5.28.2.27 get_prob_best 69
5.28.2.28 get_random 69
5.28.2.29 get_steps 69
5.28.2.30 get_steps_for 70
5.28.2.31 get_steps_for_normalized 70
5.28.2.32 get_vector_from_array 70
5.28.2.33 inc_params_accepts 70
5.28.2.34 inc_params_accepts_for 70
5.28.2.35 inc_params_rejects 70
5.28.2.36 inc_params_rejects_for 70
5.28.2.37 reset_accept_rejects 71
5.28.2.38 set_data 71
5.28.2.39 set_minmax_for 71
5.28.2.40 set_params 71
5.28.2.41 set_params_accepts_for 71
5.28.2.42 set_params_best 71
5.28.2.43 set_params_descr_all 71
5.28.2.44 set_params_descr_for 71
5.28.2.45 set_params_for 72
5.28.2.46 set_params_rejects_for 72
5.28.2.47 set_prior 72
5.28.2.48 set_prob 72
5.28.2.49 set_prob_best 72
5.28.2.50 set_random 72
5.28.2.51 set_steps_all 72
5.28.2.52 set_steps_for 72
5.28.2.53 set_steps_for_normalized 73
5.29 src/mcmc_gettersetter.h File Reference 73
5.29.1 Function Documentation 74
5.29.1.1 get_accept_rate 74
5.29.1.2 get_accept_rate_for 74
5.29.1.3 get_accept_rate_global 74
5.29.1.4 get_n_par 75
5.29.1.5 get_next_alog_urandom 75
5.29.1.6 get_next_random_jump 75
5.29.1.7 get_next_uniform_plusminus_random 75
5.29.1.8 get_next_uniform_random 75
5.29.1.9 get_params 75
5.29.1.10 get_params_accepts_for 76
5.29.1.11 get_params_accepts_global 76
5.29.1.12 get_params_accepts_sum 76
5.29.1.13 get_params_best 76
5.29.1.14 get_params_best_for 76
5.29.1.15 get_params_descr 76
5.29.1.16 get_params_for 76

5.29.1.17 get_params_max . . . 77
5.29.1.18 get_params_max_for . . . 77
5.29.1.19 get_params_min . . . 77
5.29.1.20 get_params_min_for . . . 77
5.29.1.21 get_params_rejects_for . . . 77
5.29.1.22 get_params_rejects_global . . . 77
5.29.1.23 get_params_rejects_sum . . . 77
5.29.1.24 get_prior . . . 77
5.29.1.25 get_prob . . . 78
5.29.1.26 get_prob_best . . . 78
5.29.1.27 get_random . . . 78
5.29.1.28 get_steps . . . 78
5.29.1.29 get_steps_for . . . 78
5.29.1.30 get_steps_for_normalized . . . 78
5.29.1.31 inc_params_accepts . . . 78
5.29.1.32 inc_params_accepts_for . . . 79
5.29.1.33 inc_params_rejects . . . 79
5.29.1.34 inc_params_rejects_for . . . 79
5.29.1.35 reset_accept_rejects . . . 79
5.29.1.36 set_data . . . 79
5.29.1.37 set_minmax_for . . . 79
5.29.1.38 set_model . . . 79
5.29.1.39 set_n_par . . . 79
5.29.1.40 set_params . . . 79
5.29.1.41 set_params_accepts_for . . . 80
5.29.1.42 set_params_best . . . 80
5.29.1.43 set_params_descr_all . . . 80
5.29.1.44 set_params_descr_for . . . 80
5.29.1.45 set_params_for . . . 80
5.29.1.46 set_params_rejects_for . . . 80
5.29.1.47 set_prior . . . 80
5.29.1.48 set_prob . . . 80
5.29.1.49 set_prob_best . . . 81
5.29.1.50 set_random . . . 81
5.29.1.51 set_steps_all . . . 81
5.29.1.52 set_steps_for . . . 81
5.29.1.53 set_steps_for_normalized . . . 81
5.30 src/mcmc_internal.h File Reference . . . 81
5.30.1 Define Documentation . . . 82
5.30.1.1 abs_double . . . 82
5.30.1.2 mod_double . . . 82
5.30.2 Function Documentation . . . 82
5.30.2.1 countlines . . . 82
5.31 src/mcmc_parser.c File Reference . . . 82
5.31.1 Define Documentation . . . 83
5.31.1.1 IFDEBUGPARSER . . . 83
5.31.1.2 MAX_LINE_LENGTH . . . 83
5.31.2 Function Documentation . . . 83
5.31.2.1 mcmc_load . . . 83
5.31.2.2 mcmc_load_data . . . 83

5.31.2.3 mcmc_load_params 84
5.31.2.4 mcmc_reuse_data 84
5.31.2.5 my_strdup 84
5.32 src/mcmc_struct.h File Reference 84
5.33 src/memory.h File Reference 85
5.33.1 Define Documentation 85
5.33.1.1 FREEMSG 85
5.33.1.2 mem_calloc 85
5.33.1.3 mem_free 85
5.33.1.4 mem_malloc 85
5.33.1.5 mem_realloc 86
5.33.1.6 WITHOUT_GARBAGE_COLLECTOR 86
5.34 src/parallel_tempering.c File Reference 86
5.34.1 Define Documentation 87
5.34.1.1 ADAPT 87
5.34.1.2 BURN_IN_ITERATIONS 87
5.34.1.3 RWM 87
5.34.2 Function Documentation 87
5.34.2.1 adapt 87
5.34.2.2 calibrate_first 87
5.34.2.3 calibrate_rest 88
5.34.2.4 dump 88
5.34.2.5 prepare_and_run_sampler 88
5.34.2.6 register_signal_handlers 88
5.34.2.7 report 88
5.34.2.8 run_sampler 89
5.35 src/parallel_tempering.h File Reference 89
5.35.1 Define Documentation 89
5.35.1.1 CALIBRATION_FILE 89
5.35.1.2 DUMP_ALL_CHAINS 89
5.35.1.3 PRINT_PROB_INTERVAL 90
5.35.1.4 SKIP_CALIBRATE_ALLCHAINS 90
5.35.2 Function Documentation 90
5.35.2.1 analyse_data_probability 90
5.35.2.2 analyse_marginal_distributions 90
5.35.2.3 calibrate_first 90
5.35.2.4 calibrate_rest 91
5.35.2.5 prepare_and_run_sampler 91
5.36 src/parallel_tempering_beta.c File Reference 91
5.36.1 Function Documentation 92
5.36.1.1 calc_beta_0 92
5.36.1.2 chebyshev_beta 92
5.36.1.3 chebyshev_stepwidth 92
5.36.1.4 chebyshev_temperature 92
5.36.1.5 equidistant_beta 92
5.36.1.6 equidistant_stepwidth 92
5.36.1.7 equidistant_temperature 92
5.36.1.8 get_beta 92
5.36.1.9 get_chain_beta 93
5.36.1.10 get_swapcount 93

5.36.1.11 hot_chains . . . 93
5.36.1.12 inc_swapcount . . . 93
5.36.1.13 print_current_positions . . . 93
5.36.1.14 set_beta . . . 93
5.37 src/parallel_tempering_beta.h File Reference . . . 93
5.37.1 Define Documentation . . . 94
5.37.1.1 BETA_0_STEPWIDTH . . . 94
5.37.1.2 BETA_ALIGNMENT . . . 94
5.37.2 Function Documentation . . . 94
5.37.2.1 calc_beta_0 . . . 94
5.37.2.2 get_beta . . . 95
5.37.2.3 get_chain_beta . . . 95
5.37.2.4 get_swapcount . . . 95
5.37.2.5 inc_swapcount . . . 95
5.37.2.6 print_current_positions . . . 95
5.37.2.7 set_beta . . . 95
5.38 src/parallel_tempering_config.c File Reference . . . 95
5.38.1 Function Documentation . . . 96
5.38.1.1 read_calibration_file . . . 96
5.38.1.2 setup_chains . . . 96
5.38.1.3 write_calibration_summary . . . 96
5.38.1.4 write_calibrations_file . . . 96
5.38.1.5 write_params_file . . . 97
5.39 src/parallel_tempering_config.h File Reference . . . 97
5.39.1 Define Documentation . . . 97
5.39.1.1 CALIBRATION_FILE . . . 97
5.39.2 Function Documentation . . . 97
5.39.2.1 read_calibration_file . . . 97
5.39.2.2 setup_chains . . . 98
5.39.2.3 write_calibration_summary . . . 98
5.39.2.4 write_calibrations_file . . . 98
5.39.2.5 write_params_file . . . 98
5.40 src/parallel_tempering_interaction.c File Reference . . . 98
5.40.1 Function Documentation . . . 99
5.40.1.1 parallel_tempering_decide_swap_nonrandom . . . 99
5.40.1.2 parallel_tempering_decide_swap_now . . . 99
5.40.1.3 parallel_tempering_decide_swap_random . . . 99
5.40.1.4 tempering_interaction . . . 99
5.41 src/parallel_tempering_interaction.h File Reference . . . 99
5.41.1 Define Documentation . . . 100
5.41.1.1 RANDOMSWAP . . . 100
5.41.2 Function Documentation . . . 100
5.41.2.1 tempering_interaction . . . 100
5.42 src/parallel_tempering_run.c File Reference . . . 100
5.42.1 Function Documentation . . . 101
5.42.1.1 ctrl_c_handler . . . 101
5.42.1.2 get_duration . . . 101
5.42.1.3 get_ticks_per_second . . . 101
5.42.1.4 register_signal_handlers . . . 101
5.42.1.5 sigusr_handler . . . 101

5.42.2 Variable Documentation . . . 101
5.42.2.1 dumpflag . . . 101
5.42.2.2 run . . . 102
5.43 src/parallel_tempering_run.h File Reference . . . 102
5.43.1 Function Documentation . . . 102
5.43.1.1 get_duration . . . 102
5.43.1.2 get_ticks_per_second . . . 102
5.43.1.3 register_signal_handlers . . . 102
5.43.2 Variable Documentation . . . 102
5.43.2.1 dumpflag . . . 102
5.43.2.2 run . . . 102
5.44 src/utils.c File Reference . . . 103
5.44.1 Function Documentation . . . 103
5.44.1.1 countlines . . . 103
5.44.1.2 get_column_count . . . 103
5.44.1.3 openfile . . . 103
5.45 src/utils.h File Reference . . . 103
5.45.1 Function Documentation . . . 104
5.45.1.1 countlines . . . 104
5.45.1.2 get_column_count . . . 104
5.45.1.3 openfile . . . 104
5.46 tests/run-tests.c File Reference . . . 104
5.46.1 Function Documentation . . . 105
5.46.1.1 count_tests . . . 105
5.46.1.2 main . . . 105
5.46.1.3 test . . . 105
5.46.1.4 test_all . . . 105
5.46.1.5 usage . . . 105
5.46.2 Variable Documentation . . . 105
5.46.2.1 tests_registration . . . 105
5.47 tests/tests.c File Reference . . . 106
5.47.1 Define Documentation . . . 107
5.47.1.1 ASSERT . . . 107
5.47.1.2 ASSERTDUMP . . . 107
5.47.1.3 ASSERTEQUALD . . . 107
5.47.1.4 ASSERTEQUALI . . . 107
5.47.1.5 CIRCULAR_PARAMS . . . 108
5.47.1.6 DUMPONFAIL . . . 108
5.47.2 Function Documentation . . . 108
5.47.2.1 calc_model . . . 108
5.47.2.2 calc_model_for . . . 108
5.47.2.3 calc_prob . . . 108
5.47.2.4 count_tests . . . 108
5.47.2.5 test_append . . . 109
5.47.2.6 test_create . . . 109
5.47.2.7 test_hist . . . 109
5.47.2.8 test_load . . . 109
5.47.2.9 test_mod . . . 109
5.47.2.10 test_random . . . 109
5.47.2.11 test_tests . . . 109

5.47.2.12 test_write 109
5.47.2.13 test_write_prob 109
5.47.3 Variable Documentation 110
5.47.3.1 tests_registration 110

Chapter 1

Main Page

This is the code documentation of APEMoST, the Automated Parameter Estimation and Model Selection Toolkit.

You can find more information about it `in its documentation`.

1.1 Compile-time parameters

Instead of adjusting parameters in files (which would cause a lot of unnecessary versions), you should pass the parameters to the compiler. This includes decisions on which algorithm to use, as well as fine- tuning values like number of burn-in iterations.

Do it like this:

```
$ CCFLAGS="-DN_PARAMETERS=3 -DDEBUG" make simplesin.exe
```

The possible parameters and their values:

1.1.1 Fine-tuning the algorithm

- **BETA_ALIGNMENT** (p. 94)
- **N_BETA** (p. 34)
- **BETA_0** (p. 33)
- **BURN_IN_ITERATIONS** (p. 87)
- **TARGET_ACCEPTANCE_RATE** (p. 34)
- **MAX_AR_DEVIATION** (p. 33)
- **ITER_LIMIT** (p. 33)
- **MUL** (p. 33)
- **N_SWAP** (p. 34)

- **N_PARAMETERS** (p. 65)
- **SKIP_CALIBRATE_ALLCHAINS** (p. 90)
- **PROPOSAL** (p. 66)

1.1.2 Defining algorithm behaviour

- **RANDOMSWAP** (p. 100)
- **ADAPT** (p. 87)
- **RWM** (p. 87)

1.1.3 Running

- **MAX_ITERATIONS** (p. 16)
- **DUMP_ALL_CHAINS** (p. 89)
- **PRINT_PROB_INTERVAL** (p. 90)

1.1.4 Analyzing

- **GNUPLOT_STYLE** (p. 26)
- **HISTOGRAMS_MINMAX** (p. 26)

1.1.5 Others

- **DEBUG** (p. 30)
- **VERBOSE** (p. 32)
- **SEGV** (p. 32)
- **NOASSERT** (p. 57)
- **WITHOUT_GARBAGE_COLLECTOR** (p. 86)

1.2 Runtime parameters

At runtime, the program looks for the files **PARAMS_FILENAME** (p. 34) and **DATA_-FILENAME** (p. 33).

Really, read the manual.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

mcmc 7
parallel_tempering_mcmc 11
Problem 11

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

apps/**benchmark_main.c** 13
apps/**bernoulli_example.c** 13
apps/**eval_main.c** 15
apps/**generic_main.c** 15
apps/**library.c** 17
apps/**normal.c** 19
apps/**pulse.c** 19
apps/**pulse_vrot.c** 21
apps/**simplesin.c** 22
apps/**simplesin2.c** 23
apps/**simplesin5.c** 24
src/**analyse.c** 26
src/**analyse.h** 27
src/**debug.c** 28
src/**debug.h** 28
src/**define_defaults.h** 33
src/**gsl_helper.c** 34
src/**gsl_helper.h** 37
src/**histogram.c** 40
src/**histogram.h** 42
src/**markov_chain.c** 43
src/**markov_chain.h** 46
src/**markov_chain_calibrate.c** 50
src/**mcmc.c** 54
src/**mcmc.h** 55
src/**mcmc_calculate.c** 61
src/**mcmc_dump.c** 62
src/**mcmc_gettersetter.c** 64
src/**mcmc_gettersetter.h** 73

src/**mcmc_internal.h** 81
src/**mcmc_parser.c** 82
src/**mcmc_struct.h** 84
src/**memory.h** 85
src/**parallel_tempering.c** 86
src/**parallel_tempering.h** 89
src/**parallel_tempering_beta.c** 91
src/**parallel_tempering_beta.h** 93
src/**parallel_tempering_config.c** 95
src/**parallel_tempering_config.h** 97
src/**parallel_tempering_interaction.c** 98
src/**parallel_tempering_interaction.h** 99
src/**parallel_tempering_run.c** 100
src/**parallel_tempering_run.h** 102
src/**utils.c** 103
src/**utils.h** 103
tests/**run-tests.c** 104
tests/**tests.c** 106

Chapter 4

Data Structure Documentation

4.1 mcmc Struct Reference

`#include <mcmc_struct.h>`

Data Fields

- unsigned int **n_par**
- unsigned long **accept**
- unsigned long **reject**
- double **prob**
- double **prior**
- double **prob_best**
- gsl_rng * **random**
- gsl_vector * **params**
- gsl_vector * **params_best**
- FILE ** **files**
- const char ** **params_descr**
- unsigned long * **params_accepts**
- unsigned long * **params_rejects**
- gsl_vector * **params_step**
- gsl_vector * **params_min**
- gsl_vector * **params_max**
- const gsl_matrix * **data**
- unsigned long **n_iter**
- void * **additional_data**

4.1.1 Detailed Description

The main class of operation.

4.1.2 Field Documentation

4.1.2.1 unsigned long mcmc::accept

number of accepted steps for MCMC (after calibration)

Referenced by get_params_accepts_global(), inc_params_accepts(), mcmc_init(), and reset_accept_rejects().

4.1.2.2 void* mcmc::additional_data

any extensions can be stored here

Referenced by calibrate_rest(), get_beta(), get_swapcount(), inc_swapcount(), main(), set_beta(), and setup_chains().

4.1.2.3 const gsl_matrix* mcmc::data

arbitrary sized array containing the observation as found in the file "data"

column 0 is the x-data column 1 is the y-data etc.

Referenced by apply_formula(), calc_model(), get_data(), mcmc_check(), mcmc_dump_-y_dat(), mcmc_free(), mcmc_init(), mcmc_reuse_data(), set_data(), test_load(), and test_write().

4.1.2.4 FILE** mcmc::files

files where visited nodes are written to.

Referenced by mcmc_dump_close(), mcmc_dump_current(), mcmc_dump_flush(), mcmc_-init(), and mcmc_open_dump_files().

4.1.2.5 unsigned long mcmc::n_iter

number of iterations calculated

Referenced by mcmc_append_current_parameters(), mcmc_init(), rmw_adapt_stepwidth(), run_sampler(), and test_append().

4.1.2.6 unsigned int mcmc::n_par

number of parameters

Referenced by mcmc_check(), mcmc_init(), mcmc_open_dump_files(), set_params(), test_create(), and test_load().

4.1.2.7 gsl_vector* mcmc::params

current parameters size = n_par

Referenced by calc_model(), do_step_for(), get_params(), get_params_for(), markov_chain_step(), markov_chain_step_for(), mcmc_check(), mcmc_check_best(), mcmc_dump_current(), mcmc_free(), mcmc_init(), set_params(), set_params_for(), test_load(), and test_write_prob().

4.1.2.8 unsigned long* mcmc::params_accepts

number of accepted steps for individual parameters size = n_par

Referenced by get_accept_rate(), get_params_accepts_for(), get_params_accepts_sum(), inc_params_accepts_for(), mcmc_free(), mcmc_init(), and set_params_accepts_for().

4.1.2.9 gsl_vector* mcmc::params_best

best parameters yet size = n_par

Referenced by get_params_best(), get_params_best_for(), mcmc_check(), mcmc_free(), mcmc_init(), and set_params_best().

4.1.2.10 const char** mcmc::params_descr

descriptions of parameters size = n_par

Referenced by get_params_descr(), mcmc_free(), mcmc_init(), mcmc_open_dump_files(), set_params_descr_all(), set_params_descr_for(), and test_load().

4.1.2.11 gsl_vector* mcmc::params_max

upper limits for each parameter size = n_par

Referenced by burn_in(), do_step_for(), get_params_max(), get_params_max_for(), markov_chain_calibrate_orig(), mcmc_free(), mcmc_init(), rmw_adapt_stepwidth(), set_minmax_for(), test_load(), and test_write_prob().

4.1.2.12 gsl_vector* mcmc::params_min

lower limits for each parameter size = n_par

Referenced by burn_in(), do_step_for(), get_params_min(), get_params_min_for(), markov_chain_calibrate_orig(), mcmc_free(), mcmc_init(), rmw_adapt_stepwidth(), set_minmax_for(), and test_load().

4.1.2.13 unsigned long* mcmc::params_rejects

number of rejected steps for individual parameters size = n_par

Referenced by get_accept_rate(), get_params_rejects_for(), get_params_rejects_sum(), inc_params_rejects_for(), mcmc_free(), mcmc_init(), and set_params_rejects_for().

4.1.2.14 gsl_vector* mcmc::params_step

current step widths for individual parameters size = n_par; set by calibration

Referenced by burn_in(), calibrate_rest(), do_step_for(), get_steps(), get_steps_for(), markov_chain_calibrate_orig(), mcmc_check(), mcmc_free(), mcmc_init(), set_steps_-for(), set_steps_for_normalized(), and test_load().

4.1.2.15 double mcmc::prior

explicit prior, so it can be subtracted from prob

Referenced by get_prior(), mcmc_init(), and set_prior().

4.1.2.16 double mcmc::prob

probability of the most recently evaluated parameter values

Referenced by get_prob(), mcmc_check_best(), mcmc_init(), and set_prob().

4.1.2.17 double mcmc::prob_best

probability of best parameter values yet

Referenced by get_prob_best(), mcmc_check_best(), mcmc_init(), and set_prob_best().

4.1.2.18 gsl_rng* mcmc::random

random number generator

Referenced by get_random(), init_seed(), mcmc_free(), and set_random().

4.1.2.19 unsigned long mcmc::reject

number of rejected steps for MCMC (after calibration)

Referenced by get_params_rejects_global(), inc_params_rejects(), mcmc_init(), and reset_accept_rejects().

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

- src/**mcmc_struct.h**

4.2 parallel_tempering_mcmc Struct Reference

`#include <parallel_tempering_beta.h>`

Data Fields

- double **beta**
- unsigned long **swapcount**

4.2.1 Field Documentation

4.2.1.1 double parallel_tempering_mcmc::beta

likelihood of acceptance

4.2.1.2 unsigned long parallel_tempering_mcmc::swapcount

times this was swapped

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

- src/**parallel_tempering_beta.h**

4.3 Problem Struct Reference

Data Fields

- double(∗ **Prior**)(**mcmc** ∗m, const gsl_vector ∗old_values)
- double(∗ **LogLike**)(**mcmc** ∗m, const gsl_vector ∗old_values)

4.3.1 Field Documentation

4.3.1.1 double(∗ Problem::LogLike)(mcmc ∗m, const gsl_vector ∗old_values)

Referenced by calc_model(), and set_function().

4.3.1.2 double(∗ Problem::Prior)(mcmc ∗m, const gsl_vector ∗old_values)

Referenced by calc_model(), and set_function().

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

- apps/**library.c**

Chapter 5

File Documentation

5.1 apps/benchmark_main.c File Reference

```
#include <unistd.h>
#include <string.h>
#include "mcmc.h"
#include "debug.h"
#include "parallel_tempering.h"
#include "parallel_tempering_interaction.h"
#include "define_defaults.h"
```

Functions

- int **main** (int argc, char **argv)

5.1.1 Function Documentation

5.1.1.1 int main (int *argc,* char ** *argv*)

References mcmc::additional_data, assert, calc_model(), calc_model_for(), DATA_FILENAME, dump_d, get_n_par(), get_params(), get_prob(), mcmc_check(), mcmc_load_data(), mcmc_load_params(), mem_malloc, p, PARAMS_FILENAME, and set_beta().

5.2 apps/bernoulli_example.c File Reference

```
#include <signal.h>
```

```
#include <gsl/gsl_sf.h>
#include "mcmc.h"
#include "parallel_tempering.h"
#include "debug.h"
```

Defines

- #define **SIGMA** 2

Functions

- void **calc_model** (**mcmc** *m, const gsl_vector *old_values)
- void **calc_model_for** (**mcmc** *m, const unsigned int i, const double old_value)

5.2.1 Define Documentation

5.2.1.1 #define SIGMA 2

Referenced by calc_model().

5.2.2 Function Documentation

5.2.2.1 void calc_model (mcmc * *m*, const gsl_vector * *old_values*)

update the model according to the new parameter values and recalculate the probability for the model

Parameters

m	
old_values	previous values, or NULL

References assert, mcmc::data, get_beta(), get_n_par(), get_params_for(), set_prior(), set_prob(), and SIGMA.

5.2.2.2 void calc_model_for (mcmc * *m*, const unsigned int *i*, const double *old_value*)

update the model as the new parameter value i changed and recalculate the probability for the model

Parameters

m	
i	index of the parameter value that changed
old_value	previous value of the parameter

References calc_model().

5.3 apps/eval_main.c File Reference

```
#include <unistd.h>
#include <string.h>
#include "mcmc.h"
#include "debug.h"
#include "parallel_tempering.h"
#include "parallel_tempering_interaction.h"
#include "define_defaults.h"
```

Functions

- int **main** (int argc, char **argv)

5.3.1 Function Documentation

5.3.1.1 int main (int *argc,* char ** *argv*)

References mcmc::additional_data, assert, calc_model(), DATA_FILENAME, DUMP_-FORMAT, get_n_par(), get_params_max_for(), get_params_min_for(), get_prior(), get_-prob(), mcmc_check(), mcmc_load_data(), mcmc_load_params(), mem_malloc, PARAMS_-FILENAME, r, set_beta(), and set_params_for().

5.4 apps/generic_main.c File Reference

```
#include <unistd.h>
#include <string.h>
#include "mcmc.h"
#include "debug.h"
#include "parallel_tempering.h"
#include "parallel_tempering_interaction.h"
#include "define_defaults.h"
```

Defines

- #define **MAX_ITERATIONS** 0

- #define **OUTPUT_PARAMD**(P) printf("\t%s: %f\n", #P, P);
- #define **OUTPUT_PARAMI**(P) printf("\t%s: %d\n", #P, P);

Functions

- void **usage** ()
- void **help_phase** (char ∗phase)
- void **check** ()
- int **main** (int argc, char ∗∗argv)
- void **checkfile** (char ∗filename)

Variables

- char ∗ **progname**

5.4.1 Define Documentation

5.4.1.1 #define MAX_ITERATIONS 0

set the number of iterations after you want the program to terminate.

This is especially useful in benchmarking. Example: Set this to 100000.

0 means run indefinitely

Referenced by check(), and main().

5.4.1.2 #define OUTPUT_PARAMD(*P*) printf("\t%s: %f\n", #P, P);

Referenced by check().

5.4.1.3 #define OUTPUT_PARAMI(*P*) printf("\t%s: %d\n", #P, P);

Referenced by check().

5.4.2 Function Documentation

5.4.2.1 void check ( )

References BETA_0, BETA_ALIGNMENT, BURN_IN_ITERATIONS, checkfile(), DATA_FILENAME, ITER_LIMIT, MAX_AR_DEVIATION, MAX_ITERATIONS, MUL, N_BETA, N_PARAMETERS, N_SWAP, OUTPUT_PARAMD, OUTPUT_PARAMI, PARAMS_FILENAME, PRINT_PROB_INTERVAL, progname, TARGET_ACCEPTANCE_-RATE, and TOSTRING.

Referenced by main().

5.4.2.2 void checkfile (char * *filename*)

Referenced by check().

5.4.2.3 void help_phase (char * *phase*)

References CALIBRATION_FILE, DATA_FILENAME, PARAMS_FILENAME, and usage().

Referenced by main().

5.4.2.4 int main (int *argc,* char ** *argv*)

References analyse_data_probability(), analyse_marginal_distributions(), calibrate_-first(), calibrate_rest(), check(), help_phase(), MAX_ITERATIONS, prepare_and_run_-sampler(), progname, and usage().

5.4.2.5 void usage ( )

References progname.

Referenced by help_phase(), and main().

5.4.3 Variable Documentation

5.4.3.1 char* progname

Referenced by check(), main(), and usage().

5.5 apps/library.c File Reference

```
#include <gsl/gsl_sf.h>
#include "mcmc.h"
#include "parallel_tempering.h"
```

Data Structures

- struct **Problem**

Functions

- void **set_function** (double(*LogLike)(**mcmc** *m, const gsl_vector *old_values), double(*Prior)(**mcmc** *m, const gsl_vector *old_values))

- void **calc_model** (**mcmc** ∗m, const gsl_vector ∗old_values)
- void **calc_model_for** (**mcmc** ∗m, const unsigned int i, const double old_value)

Variables

- struct **Problem p**

5.5.1 Function Documentation

5.5.1.1 void calc_model (mcmc ∗ *m,* const gsl_vector ∗ *old_values*)

update the model according to the new parameter values and recalculate the probability for the model

Parameters

m	
old_values	previous values, or NULL

References get_beta(), get_prior(), Problem::LogLike, p, Problem::Prior, set_prior(), and set_prob().

5.5.1.2 void calc_model_for (mcmc ∗ *m,* const unsigned int *i,* const double *old_value*)

update the model as the new parameter value i changed and recalculate the probability for the model

Parameters

m	
i	index of the parameter value that changed
old_value	previous value of the parameter

References calc_model().

5.5.1.3 void set_function (double(∗)(mcmc ∗m, const gsl_vector ∗old_values) *LogLike,* double(∗)(mcmc ∗m, const gsl_vector ∗old_values) *Prior*)

References Problem::LogLike, p, and Problem::Prior.

5.5.2 Variable Documentation

5.5.2.1 struct Problem p

Referenced by calc_model(), linreg_n(), main(), and set_function().

5.6 apps/normal.c File Reference

```
#include <signal.h>
#include <gsl/gsl_sf.h>
#include "mcmc.h"
#include "parallel_tempering.h"
#include "debug.h"
```

Functions

- void **calc_model** (**mcmc** *m, const gsl_vector *old_values)
- void **calc_model_for** (**mcmc** *m, const unsigned int i, const double old_value)

5.6.1 Function Documentation

5.6.1.1 void calc_model (mcmc * *m*, const gsl_vector * *old_values*)

update the model according to the new parameter values and recalculate the probability for the model

Parameters

m	
old_values	previous values, or NULL

References get_beta(), mcmc::params, and set_prob().

5.6.1.2 void calc_model_for (mcmc * *m*, const unsigned int *i*, const double *old_value*)

update the model as the new parameter value i changed and recalculate the probability for the model

Parameters

m	
i	index of the parameter value that changed
old_value	previous value of the parameter

References calc_model().

5.7 apps/pulse.c File Reference

```
#include <signal.h>
#include <gsl/gsl_sf.h>
```

```
#include "mcmc.h"
#include "parallel_tempering.h"
#include "debug.h"
```

Defines

- #define **HMIN** 1e-6

Functions

- void **calc_model** (**mcmc** *m, const gsl_vector *old_values)
- void **calc_model_for** (**mcmc** *m, const unsigned int j, const double old_value)

5.7.1 Define Documentation

5.7.1.1 #define HMIN 1e-6

5.7.2 Function Documentation

5.7.2.1 void calc_model (mcmc * *m,* const gsl_vector * *old_values*)

update the model according to the new parameter values and recalculate the probability for the model

Parameters

m	
old_values	previous values, or NULL

References assert, mcmc::data, get_beta(), get_n_par(), get_prior(), mcmc::params, set_prior(), and set_prob().

5.7.2.2 void calc_model_for (mcmc * *m,* const unsigned int *i,* const double *old_value*)

update the model as the new parameter value i changed and recalculate the probability for the model

Parameters

m	
i	index of the parameter value that changed
old_value	previous value of the parameter

References calc_model().

5.8 apps/pulse_vrot.c File Reference

```
#include <signal.h>
#include <gsl/gsl_sf.h>
#include "mcmc.h"
#include "parallel_tempering.h"
#include "debug.h"
```

Defines

- #define **HMIN** 1e-6

Functions

- void **calc_model** (**mcmc** *m, const gsl_vector *old_values)
- void **calc_model_for** (**mcmc** *m, const unsigned int j, const double old_value)

5.8.1 Define Documentation

5.8.1.1 #define HMIN 1e-6

5.8.2 Function Documentation

5.8.2.1 void calc_model (mcmc * *m*, const gsl_vector * *old_values*)

update the model according to the new parameter values and recalculate the probability for the model

Parameters

m	
old_values	previous values, or NULL

References assert, mcmc::data, get_beta(), get_n_par(), get_prior(), mcmc::params, set_prior(), and set_prob().

5.8.2.2 void calc_model_for (mcmc * *m*, const unsigned int *i*, const double *old_value*)

update the model as the new parameter value i changed and recalculate the probability for the model

Parameters

m	
i	index of the parameter value that changed
old_value	previous value of the parameter

References calc_model().

5.9 apps/simplesin.c File Reference

```
#include <signal.h>
#include <gsl/gsl_sf.h>
#include "mcmc.h"
#include "parallel_tempering.h"
#include "debug.h"
```

Defines

- #define **SIGMA** 0.5

Functions

- double **apply_formula** (**mcmc** *m, unsigned int i, double amplitude, double frequency, double phase, double offset)
- void **calc_model** (**mcmc** *m, const gsl_vector *old_values)
- void **calc_model_for** (**mcmc** *m, const unsigned int i, const double old_value)

5.9.1 Define Documentation

5.9.1.1 #define SIGMA 0.5

Referenced by calc_model().

5.9.2 Function Documentation

5.9.2.1 double apply_formula (mcmc * *m,* unsigned int *i,* double *amplitude,* double *frequency,* double *phase,* double *offset*)

References mcmc::data.

Referenced by calc_model().

5.9.2.2 void calc_model (mcmc * *m,* const gsl_vector * *old_values*)

update the model according to the new parameter values and recalculate the probability for the model

Parameters

m	
old_values	previous values, or NULL

References apply_formula(), mcmc::data, get_beta(), mcmc::params, set_prob(), and SIGMA.

5.9.2.3 void calc_model_for (mcmc * *m*, const unsigned int *i*, const double *old_value*)

update the model as the new parameter value i changed and recalculate the probability for the model

Parameters

m	
i	index of the parameter value that changed
old_value	previous value of the parameter

References calc_model().

5.10 apps/simplesin2.c File Reference

```
#include <signal.h>
#include <gsl/gsl_sf.h>
#include "mcmc.h"
#include "parallel_tempering.h"
#include "debug.h"
```

Defines

- #define **SIGMA** 0.5

Functions

- double **apply_formula** (**mcmc** *m, unsigned int i, double param0, double param1)
- void **calc_model** (**mcmc** *m, const gsl_vector *old_values)
- void **calc_model_for** (**mcmc** *m, const unsigned int i, const double old_value)

5.10.1 Define Documentation

5.10.1.1 #define SIGMA 0.5

Referenced by calc_model().

5.10.2 Function Documentation

5.10.2.1 double apply_formula (mcmc * *m,* unsigned int *i,* double *param0,* double *param1*)

References mcmc::data.

5.10.2.2 void calc_model (mcmc * *m,* const gsl_vector * *old_values*)

update the model according to the new parameter values and recalculate the probability for the model

Parameters

m	
old_values	previous values, or NULL

References apply_formula(), mcmc::data, get_beta(), mcmc::params, set_prob(), and SIGMA.

5.10.2.3 void calc_model_for (mcmc * *m,* const unsigned int *i,* const double *old_value*)

update the model as the new parameter value i changed and recalculate the probability for the model

Parameters

m	
i	index of the parameter value that changed
old_value	previous value of the parameter

References calc_model().

5.11 apps/simplesin5.c File Reference

```
#include <signal.h>
#include <gsl/gsl_sf.h>
#include "mcmc.h"
#include "parallel_tempering.h"
#include "debug.h"
```

Defines

- #define **SIGMA** 0.5

Functions

- double **apply_formula** (**mcmc** ∗m, unsigned int i, double param0, double param1, double param2)
- void **calc_model** (**mcmc** ∗m, const gsl_vector ∗old_values)
- void **calc_model_for** (**mcmc** ∗m, const unsigned int i, const double old_value)

5.11.1 Define Documentation

5.11.1.1 #define SIGMA 0.5

TODO: document

Referenced by calc_model().

5.11.2 Function Documentation

5.11.2.1 double apply_formula (mcmc ∗ *m,* unsigned int *i,* double *param0,* double *param1,* double *param2*)

References mcmc::data.

5.11.2.2 void calc_model (mcmc ∗ *m,* const gsl_vector ∗ *old_values*)

update the model according to the new parameter values and recalculate the probability for the model

Parameters

m	
old_values	previous values, or NULL

References apply_formula(), mcmc::data, get_beta(), mcmc::params, set_prob(), and SIGMA.

5.11.2.3 void calc_model_for (mcmc ∗ *m,* const unsigned int *i,* const double *old_value*)

update the model as the new parameter value i changed and recalculate the probability for the model

Parameters

m	
i	index of the parameter value that changed
old_value	previous value of the parameter

References calc_model().

5.12 src/analyse.c File Reference

```
#include <omp.h>
#include "mcmc.h"
#include "parallel_tempering_config.h"
#include "debug.h"
#include "define_defaults.h"
#include "gsl_helper.h"
#include "parallel_tempering_run.h"
#include "histogram.h"
#include "utils.h"
```

Defines

- #define **NBINS** 200
- #define **HISTOGRAMS_MINMAX**
- #define **GNUPLOT_STYLE** "with histeps"

Functions

- void **analyse_data_probability** ()
- double **calc_mcmc_error** (const double mean, const char *filename, unsigned long batchsize)
- void **calc_marginal_distribution** (**mcmc** **chains, unsigned int n_beta, unsigned int param, int find_minmax)
- void **analyse_marginal_distributions** ()

5.12.1 Define Documentation

5.12.1.1 #define GNUPLOT_STYLE "with histeps"

Referenced by analyse_marginal_distributions().

5.12.1.2 #define HISTOGRAMS_MINMAX

If not set, the marginal distribution will be calculated for the whole parameter range. Pros: faster, comparable. Cons: Not as detailed.

If set, the maximum and minimum values found are used for the histogram. Pros: more detailed in the area of interest

5.12.1.3 #define NBINS 200

Referenced by calc_marginal_distribution().

5.12.2 Function Documentation

5.12.2.1 void analyse_data_probability ( )

References assert, dump_s, get_beta(), N_BETA, read_calibration_file(), and setup_chains().

Referenced by main().

5.12.2.2 void analyse_marginal_distributions ( )

References assert, calc_marginal_distribution(), get_n_par(), get_params_descr(), GNUPLOT_STYLE, N_BETA, read_calibration_file(), and setup_chains().

Referenced by main().

5.12.2.3 void calc_marginal_distribution (mcmc ** *chains*, unsigned int *n_beta*, unsigned int *param*, int *find_minmax*)

References append_to_hists(), assert, calc_mcmc_error(), create_hist(), debug, DUMP_FORMAT, dump_s, dump_v, find_min_max(), get_column_count(), get_params_descr(), get_params_max_for(), get_params_min_for(), NBINS, and update_min_max().

Referenced by analyse_marginal_distributions().

5.12.2.4 double calc_mcmc_error (const double *mean*, const char * *filename*, unsigned long *batchsize*)

References openfile().

Referenced by calc_marginal_distribution().

5.13 src/analyse.h File Reference

```
#include "mcmc.h"
```

Functions

- void **analyse_marginal_distributions** ()
- void **analyse_data_probability** ()

5.13.1 Function Documentation

5.13.1.1 void analyse_data_probability ( )

5.13.1.2 void analyse_marginal_distributions ( )

5.14 src/debug.c File Reference

```
#include "debug.h"
```

Functions

- void **dump_vector** (const gsl_vector ∗v)
- void **dump_vectorln** (const gsl_vector ∗v)
- void **dump_mcmc** (const **mcmc** ∗m)

5.14.1 Function Documentation

5.14.1.1 void dump_mcmc (const mcmc ∗ *m*)

Dump the mcmc structure

References debug, dump_i_s, dump_p, dump_ul, dump_v, get_n_par(), IFDEBUG, and IFSEGV.

Referenced by dump(), and test_create().

5.14.1.2 void dump_vector (const gsl_vector ∗ *v*)

Referenced by dump(), and dump_vectorln().

5.14.1.3 void dump_vectorln (const gsl_vector ∗ *v*)

References dump_vector().

Referenced by calibrate_rest(), and print_current_positions().

5.15 src/debug.h File Reference

```
#include <stdio.h>
#include "memory.h"
#include "mcmc.h"
```

Defines

- #define **DEBUG**
- #define **IFDEBUG** if(1)
- #define **SEGV**
- #define **IFSEGV** if(1)
- #define **IFWAIT** if(0)
- #define **VERBOSE**
- #define **IFVERBOSE** if(1)
- #define **STRINGIFY**(x) #x
- #define **TOSTRING**(x) STRINGIFY(x)
- #define **AT** __FILE__ ":" TOSTRING(__LINE__)
- #define **debug**(str) IFDEBUG { printf("\tDEBUG[%s]: %s\n", AT, str); fflush(NULL); }
- #define **dump_i**(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %i\n", AT, str, var); fflush(NULL); }
- #define **dump_ui**(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %u\n", AT, str, var); fflush(NULL); }
- #define **dump_d**(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %f\n", AT, str, var); fflush(NULL); }
- #define **dump_l**(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %l\n", AT, str, var); fflush(NULL); }
- #define **dump_ul**(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %lu\n", AT, str, var); fflush(NULL); }
- #define **dump_size**(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %lu\n", AT, str, (unsigned long)var); fflush(NULL); }
- #define **dump_i_s**(str, index, var) IFDEBUG { printf("\tDEBUG[%s]: %s[%i]: %s\n", AT, str, index, var); fflush(NULL); }
- #define **dump_s**(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %s\n", AT, str, var); fflush(NULL); }
- #define **dump_p**(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %**p**\n", AT, str, var); fflush(NULL); }
- #define **dump_v**(str, v) IFDEBUG { printf("\tDEBUG[%s]: %s: ", AT, str); dump_vectorln(v); fflush(NULL); }
- #define **dump_m**(str, m) IFDEBUG { printf("\tDEBUG[%s]: %s\n", AT, str); dump(m); fflush(NULL); }
- #define **require**(x) (x)

Functions

- void **dump_mcmc** (const **mcmc** *m)
- void **dump_vector** (const gsl_vector *v)
- void **dump_vectorln** (const gsl_vector *v)

5.15.1 Define Documentation

5.15.1.1 #define AT __FILE__ ":" TOSTRING(__LINE__)

5.15.1.2 #define DEBUG

Turns on debug output.

One could think that turning this off would improve performance, test have shown the additional call time is not significant.

5.15.1.3 #define debug(*str*) IFDEBUG { printf("\tDEBUG[%s]: %s\n", AT, str); fflush(NULL); }

Referenced by burn_in(), calc_hist(), calc_marginal_distribution(), dump(), dump_mcmc(), markov_chain_calibrate_linear_regression(), markov_chain_calibrate_orig(), mcmc_free(), mcmc_init(), mcmc_reuse_data(), parallel_tempering_decide_swap_random(), test_create(), test_write(), and test_write_prob().

5.15.1.4 #define dump_d(*str, var*) IFDEBUG { printf("\tDEBUG[%s]: %s: %f\n", AT, str, var); fflush(NULL); }

Referenced by calc_hist(), main(), markov_chain_calibrate(), and markov_chain_calibrate_orig().

5.15.1.5 #define dump_i(*str, var*) IFDEBUG { printf("\tDEBUG[%s]: %s: %i\n", AT, str, var); fflush(NULL); }

Referenced by adapt(), append_to_hists(), find_min_max(), markov_chain_calibrate_orig(), and mcmc_load_params().

5.15.1.6 #define dump_i_s(*str, index, var*) IFDEBUG { printf("\tDEBUG[%s]: %s[%i]: %s\n", AT, str, index, var); fflush(NULL); }

Referenced by dump_mcmc().

5.15.1.7 #define dump_l(*str, var*) IFDEBUG { printf("\tDEBUG[%s]: %s: %l\n", AT, str, var); fflush(NULL); }

5.15.1.8 #define dump_m(*str, m*) IFDEBUG { printf("\tDEBUG[%s]: %s\n", AT, str); dump(m); fflush(NULL); }

5.15.1.9 #define dump_p(*str, var*) IFDEBUG { printf("\tDEBUG[%s]: %s: %p\n", AT, str, var); fflush(NULL); }

Referenced by dump_mcmc().

5.15.1.10 #define dump_s(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %s\n", AT, str, var); fflush(NULL); }

Referenced by analyse_data_probability(), calc_marginal_distribution(), mcmc_load_data(), and mcmc_open_dump_files().

5.15.1.11 #define dump_size(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %lu\n", AT, str, (unsigned long)var); fflush(NULL); }

5.15.1.12 #define dump_ui(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %u\n", AT, str, var); fflush(NULL); }

5.15.1.13 #define dump_ul(str, var) IFDEBUG { printf("\tDEBUG[%s]: %s: %lu\n", AT, str, var); fflush(NULL); }

Referenced by burn_in(), dump(), dump_mcmc(), and markov_chain_calibrate_orig().

5.15.1.14 #define dump_v(str, v) IFDEBUG { printf("\tDEBUG[%s]: %s: ", AT, str); dump_vectorln(v); fflush(NULL); }

Referenced by burn_in(), calc_marginal_distribution(), dump_mcmc(), markov_chain_calibrate_alt(), markov_chain_calibrate_multilinear_regression(), markov_chain_calibrate_orig(), and mcmc_check_best().

5.15.1.15 #define IFDEBUG if(1)

Referenced by dump(), dump_mcmc(), linreg_n(), markov_chain_calibrate_linear_regression(), markov_chain_calibrate_multilinear_regression(), markov_chain_calibrate_orig(), markov_chain_calibrate_quadratic(), and sort().

5.15.1.16 #define IFSEGV if(1)

Referenced by dump_mcmc(), mcmc_free(), and mcmc_init().

5.15.1.17 #define IFVERBOSE if(1)

Referenced by assess_acceptance_rate(), burn_in(), do_step_for(), mcmc_open_dump_files(), and parallel_tempering_decide_swap_random().

5.15.1.18 #define IFWAIT if(0)

5.15.1.19 #define require(x) (x)

Referenced by calc_hist(), calc_normalized(), dup_vector(), and test_write_prob().

5.15.1.20 #define SEGV

Turns on debug output for allocation and free() to track segfaults.

You can also use libduma or valgrind:

DUMA

LD_PRELOAD=libduma.so.0.0.0 yourprogram

valgrind

valgrind --tool=memcheck --leak-check=full yourprogram

5.15.1.21 #define STRINGIFY(*x*) #x

5.15.1.22 #define TOSTRING(*x*) STRINGIFY(x)

Referenced by check().

5.15.1.23 #define VERBOSE

additionally to DEBUG, print more.

5.15.2 Function Documentation

5.15.2.1 void dump_mcmc (const mcmc * *m*)

Dump the mcmc structure

References debug, dump_i_s, dump_p, dump_ul, dump_v, get_n_par(), IFDEBUG, and IFSEGV.

Referenced by dump(), and test_create().

5.15.2.2 void dump_vector (const gsl_vector * *v*)

Referenced by dump(), and dump_vectorln().

5.15.2.3 void dump_vectorln (const gsl_vector * *v*)

References dump_vector().

Referenced by calibrate_rest(), and print_current_positions().

5.16 src/define_defaults.h File Reference

Defines

- #define **N_BETA** 20
- #define **BETA_0** -0.001
- #define **ITER_LIMIT** 100000
- #define **MUL** 0.85
- #define **N_SWAP** -30
- #define **PARAMS_FILENAME** "params"
- #define **DATA_FILENAME** "data"
- #define **TARGET_ACCEPTANCE_RATE** 0.50
- #define **MAX_AR_DEVIATION** 0.01

5.16.1 Define Documentation

5.16.1.1 #define BETA_0 -0.001

Position where the smallest/hottest chain should start. Remember, beta = 1/T, so beta = 0 is infinitely hot.

Has to be between 0 and 1. If < 0, it is determined automatically so that the hottest chain will aim at a stepwidth of 1/3 of parameter space.

Referenced by calibrate_rest(), and check().

5.16.1.2 #define DATA_FILENAME "data"

Referenced by check(), help_phase(), main(), and setup_chains().

5.16.1.3 #define ITER_LIMIT 100000

How many iterations should be used for the burn-in phase? How many iterations should be used after burn-in for adjusting the step widths?

Referenced by calibrate_first(), calibrate_rest(), and check().

5.16.1.4 #define MAX_AR_DEVIATION 0.01

How much deviation from the desired acceptance rate is acceptable

Referenced by calibrate_first(), calibrate_rest(), and check().

5.16.1.5 #define MUL 0.85

Factor used for scaling the step widths

Referenced by calibrate_first(), calibrate_rest(), and check().

5.16.1.6 #define N_BETA 20

Number of chains to use for parallel tempering

<= 12 is not recommended

Referenced by analyse_data_probability(), analyse_marginal_distributions(), calibrate_rest(), check(), prepare_and_run_sampler(), and setup_chains().

5.16.1.7 #define N_SWAP -30

After how many iterations should a swap occur?

Performance evaluations suggest to set this to 2000/N_BETA. If < 0, this will be done for you.

Referenced by check(), and prepare_and_run_sampler().

5.16.1.8 #define PARAMS_FILENAME "params"

Referenced by check(), help_phase(), main(), setup_chains(), and write_params_file().

5.16.1.9 #define TARGET_ACCEPTANCE_RATE 0.50

Desired acceptance rate after calibration

Referenced by adapt(), calibrate_first(), calibrate_rest(), check(), markov_chain_calibrate_orig(), and rmw_adapt_stepwidth().

5.17 src/gsl_helper.c File Reference

```
#include "gsl_helper.h"
#include "utils.h"
#include "debug.h"
#include <gsl/gsl_linalg.h>
```

Functions

- double **calc_vector_sum** (const gsl_vector *v)
- double **calc_vector_squaresum** (const gsl_vector *v)
- gsl_vector * **dup_vector** (const gsl_vector *v)
- gsl_vector * **calc_normalized** (const gsl_vector *v)
- int **calc_same** (const gsl_vector *a, const gsl_vector *b)
- void **max_vector** (gsl_vector *a, const gsl_vector *b)
- void **min_vector** (gsl_vector *a, const gsl_vector *b)
- void **sort** (gsl_vector **vs, unsigned int nvectors, unsigned int vector_size)

- double **min_column** (const gsl_matrix *m, const unsigned int i)
- double **min_row** (const gsl_matrix *m, const unsigned int i)
- double **max_column** (const gsl_matrix *m, const unsigned int i)
- double **max_row** (const gsl_matrix *m, const unsigned int i)
- double **xbar** (const gsl_vector *x)
- double **xbar_j** (const gsl_matrix *x, const unsigned int j)
- gsl_vector * **linreg_n** (const gsl_matrix *x, const gsl_vector *y, double *d, const gsl_vector *weights)
- double **calc_deviation** (const gsl_matrix *x, const gsl_vector *y, const gsl_vector *k, const double d, const gsl_vector *weights)

5.17.1 Function Documentation

5.17.1.1 double calc_deviation (const gsl_matrix * *x*, const gsl_vector * *y*, const gsl_vector * *k*, const double *d*, const gsl_vector * *weights*)

n-dimensional weighted square deviation

References assert.

Referenced by markov_chain_calibrate_multilinear_regression().

5.17.1.2 gsl_vector* calc_normalized (const gsl_vector * *v*)

normalizes the vector, i.e. the values are scaled so that the sum of all values is 1

The caller has to free the returned vector.

References calc_vector_sum(), dup_vector(), r, and require.

5.17.1.3 int calc_same (const gsl_vector * *a*, const gsl_vector * *b*)

Returns

1 if vectors contain the same entries

References assert.

5.17.1.4 double calc_vector_squaresum (const gsl_vector * *v*)

sums the squared values

5.17.1.5 double calc_vector_sum (const gsl_vector * *v*)

sums the values

Referenced by calc_normalized().

5.17.1.6 gsl_vector* dup_vector (const gsl_vector * v)

returns a duplicate.

The caller has to free the returned vector.

References assert, r, and require.

Referenced by burn_in(), calc_beta_0(), calc_normalized(), calibrate_rest(), markov_-chain_step(), restart_from_best(), update_min_max(), and write_calibration_summary().

5.17.1.7 gsl_vector* linreg_n (const gsl_matrix * x, const gsl_vector * y, double * d, const gsl_vector * weights)

n-dimensional weighted linear regression returns k and d.

References assert, IFDEBUG, p, xbar(), and xbar_j().

Referenced by markov_chain_calibrate_multilinear_regression().

5.17.1.8 double max_column (const gsl_matrix * m, const unsigned int i)

given the first index of the matrix, iterate through the second to find the largest entry

Referenced by markov_chain_calibrate_quadratic().

5.17.1.9 double max_row (const gsl_matrix * m, const unsigned int i)

given the second index of the matrix, iterate through the first to find the largest entry

5.17.1.10 void max_vector (gsl_vector * a, const gsl_vector * b)

a = max(a, b)

References assert.

Referenced by update_min_max().

5.17.1.11 double min_column (const gsl_matrix * m, const unsigned int i)

given the first index of the matrix, iterate through the second to find the smallest entry

Referenced by markov_chain_calibrate_quadratic().

5.17.1.12 double min_row (const gsl_matrix * m, const unsigned int i)

given the second index of the matrix, iterate through the first to find the smallest entry

5.17.1.13 void min_vector (gsl_vector * *a,* const gsl_vector * *b*)

a = min(a, b)

References assert.

Referenced by update_min_max().

5.17.1.14 void sort (gsl_vector ** *vs,* unsigned int *nvectors,* unsigned int *vector_size*)

sorts all vectors by the entries in the first vector.

selection sort

References IFDEBUG.

5.17.1.15 double xbar (const gsl_vector * *x*)

Referenced by linreg_n().

5.17.1.16 double xbar_j (const gsl_matrix * *x,* const unsigned int *j*)

Referenced by linreg_n().

5.18 src/gsl_helper.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_histogram.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_matrix.h>
#include <assert.h>
```

Functions

- double **calc_vector_sum** (const gsl_vector *v)
- double **calc_vector_squaresum** (const gsl_vector *v)
- gsl_vector * **dup_vector** (const gsl_vector *v)
- gsl_vector * **calc_normalized** (const gsl_vector *v)
- int **calc_same** (const gsl_vector *a, const gsl_vector *b)
- void **max_vector** (gsl_vector *a, const gsl_vector *b)
- void **min_vector** (gsl_vector *a, const gsl_vector *b)
- void **sort** (gsl_vector **vs, unsigned int nvectors, unsigned int vector_size)

- double **min_column** (const gsl_matrix ∗m, const unsigned int i)
- double **min_row** (const gsl_matrix ∗m, const unsigned int i)
- double **max_column** (const gsl_matrix ∗m, const unsigned int i)
- double **max_row** (const gsl_matrix ∗m, const unsigned int i)
- double **calc_deviation** (const gsl_matrix ∗x, const gsl_vector ∗y, const gsl_-vector ∗k, const double d, const gsl_vector ∗weights)
- gsl_vector ∗ **linreg_n** (const gsl_matrix ∗x, const gsl_vector ∗y, double ∗d, const gsl_vector ∗weights)

5.18.1 Function Documentation

5.18.1.1 double calc_deviation (const gsl_matrix ∗ *x*, const gsl_vector ∗ *y*, const gsl_vector ∗ *k*, const double *d*, const gsl_vector ∗ *weights*)

n-dimensional weighted square deviation

References assert.

Referenced by markov_chain_calibrate_multilinear_regression().

5.18.1.2 gsl_vector∗ calc_normalized (const gsl_vector ∗ *v*)

normalizes the vector, i.e. the values are scaled so that the sum of all values is 1

The caller has to free the returned vector.

References calc_vector_sum(), dup_vector(), r, and require.

5.18.1.3 int calc_same (const gsl_vector ∗ *a*, const gsl_vector ∗ *b*)

Returns

1 if vectors contain the same entries

References assert.

5.18.1.4 double calc_vector_squaresum (const gsl_vector ∗ *v*)

sums the squared values

5.18.1.5 double calc_vector_sum (const gsl_vector ∗ *v*)

sums the values

Referenced by calc_normalized().

5.18.1.6 gsl_vector* dup_vector (const gsl_vector * v)

returns a duplicate.

The caller has to free the returned vector.

References assert, r, and require.

Referenced by burn_in(), calc_beta_0(), calc_normalized(), calibrate_rest(), markov_-chain_step(), restart_from_best(), update_min_max(), and write_calibration_summary().

5.18.1.7 gsl_vector* linreg_n (const gsl_matrix * x, const gsl_vector * y, double * d, const gsl_vector * weights)

n-dimensional weighted linear regression returns k and d.

References assert, IFDEBUG, p, xbar(), and xbar_j().

Referenced by markov_chain_calibrate_multilinear_regression().

5.18.1.8 double max_column (const gsl_matrix * m, const unsigned int i)

given the first index of the matrix, iterate through the second to find the largest entry

Referenced by markov_chain_calibrate_quadratic().

5.18.1.9 double max_row (const gsl_matrix * m, const unsigned int i)

given the second index of the matrix, iterate through the first to find the largest entry

5.18.1.10 void max_vector (gsl_vector * a, const gsl_vector * b)

a = max(a, b)

References assert.

Referenced by update_min_max().

5.18.1.11 double min_column (const gsl_matrix * m, const unsigned int i)

given the first index of the matrix, iterate through the second to find the smallest entry

Referenced by markov_chain_calibrate_quadratic().

5.18.1.12 double min_row (const gsl_matrix * m, const unsigned int i)

given the second index of the matrix, iterate through the first to find the smallest entry

5.18.1.13 void min_vector (gsl_vector * *a*, const gsl_vector * *b*)

a = min(a, b)

References assert.

Referenced by update_min_max().

5.18.1.14 void sort (gsl_vector ** *vs*, unsigned int *nvectors*, unsigned int *vector_size*)

sorts all vectors by the entries in the first vector.

selection sort

References IFDEBUG.

5.19 src/histogram.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_histogram.h>
#include <gsl/gsl_vector.h>
#include <assert.h>
#include <string.h>
#include <ctype.h>
#include "gsl_helper.h"
#include "utils.h"
#include "debug.h"
#include "histogram.h"
```

Functions

- gsl_histogram * **create_hist** (int nbins, double min, double max)
- gsl_histogram * **calc_hist** (const gsl_vector *v, int nbins)
- void **append_to_hists** (gsl_histogram **hists, unsigned int n, const char *filename)
- void **find_min_max** (char *filename, gsl_vector *min, gsl_vector *max)
- void **update_min_max** (char *filename, gsl_vector *min, gsl_vector *max)

5.19.1 Function Documentation

5.19.1.1 void append_to_hists (gsl_histogram ** *hists*, unsigned int *n*, const char * *filename*)

append the input of a file with n columns to the according histograms.

References dump_i, and openfile().

Referenced by calc_marginal_distribution().

5.19.1.2 gsl_histogram* calc_hist (const gsl_vector * *v*, int *nbins*)

calculate a histogram

Parameters

v	vector to look at
nbins	number of bins to use for the histogram

References debug, dump_d, and require.

Referenced by test_hist().

5.19.1.3 gsl_histogram* create_hist (int *nbins*, double *min*, double *max*)

create a inclusive histogram where min and max are inside the value range.

Referenced by calc_marginal_distribution().

5.19.1.4 void find_min_max (char * *filename*, gsl_vector * *min*, gsl_vector * *max*)

find the smallest and largest values of a file for each column, then set the min/max vectors

References assert, dump_i, and openfile().

Referenced by calc_marginal_distribution(), and update_min_max().

5.19.1.5 void update_min_max (char * *filename*, gsl_vector * *min*, gsl_vector * *max*)

find the smallest and largest values of a file for each column, then update the min/max vectors

References dup_vector(), find_min_max(), max_vector(), and min_vector().

Referenced by calc_marginal_distribution().

5.20 src/histogram.h File Reference

Functions

- gsl_histogram * **create_hist** (int nbins, double min, double max)
- gsl_histogram * **calc_hist** (const gsl_vector *v, int nbins)
- void **append_to_hists** (gsl_histogram **hists, unsigned int n, const char *filename)
- void **find_min_max** (char *filename, gsl_vector *min, gsl_vector *max)
- void **update_min_max** (char *filename, gsl_vector *min, gsl_vector *max)

5.20.1 Function Documentation

5.20.1.1 void append_to_hists (gsl_histogram ** *hists*, unsigned int *n*, const char * *filename*)

append the input of a file with n columns to the according histograms.

References dump_i, and openfile().

Referenced by calc_marginal_distribution().

5.20.1.2 gsl_histogram* calc_hist (const gsl_vector * *v*, int *nbins*)

calculate a histogram

Parameters

v	vector to look at
nbins	number of bins to use for the histogram

References debug, dump_d, and require.

Referenced by test_hist().

5.20.1.3 gsl_histogram* create_hist (int *nbins*, double *min*, double *max*)

create a inclusive histogram where min and max are inside the value range.

Referenced by calc_marginal_distribution().

5.20.1.4 void find_min_max (char * *filename*, gsl_vector * *min*, gsl_vector * *max*)

find the smallest and largest values of a file for each column, then set the min/max vectors

References assert, dump_i, and openfile().

Referenced by calc_marginal_distribution(), and update_min_max().

5.20.1.5 void update_min_max (char * *filename,* gsl_vector * *min,* gsl_vector * *max*)

find the smallest and largest values of a file for each column, then update the min/max vectors

References dup_vector(), find_min_max(), max_vector(), and min_vector().

Referenced by calc_marginal_distribution().

5.21 src/markov_chain.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <libgen.h>
#include "mcmc.h"
#include "mcmc_internal.h"
#include "debug.h"
#include "gsl_helper.h"
#include <gsl/gsl_sf.h>
```

Defines

- #define **MINIMAL_STEPWIDTH** 0.0000001
- #define **MAXIMAL_STEPWIDTH** 1000000

Functions

- void **restart_from_best** (**mcmc** *m)
- void **burn_in** (**mcmc** *m, const unsigned int burn_in_iterations)
- void **clear_bit** (char *bitfield, unsigned int i)
- void **set_bit** (char *bitfield, unsigned int i)
- int **get_bit** (char *bitfield, unsigned int i)
- unsigned int **assess_acceptance_rate** (**mcmc** *m, unsigned int param, double desired_acceptance_rate, double min_accuracy, double max_accuracy, double *acceptance_rate, double *accuracy)
- void **do_step_for** (**mcmc** *m, const unsigned int i)
- void **markov_chain_step_for** (**mcmc** *m, const unsigned int index)
- void **rmw_adapt_stepwidth** (**mcmc** *m, const double prob_old)
- void **markov_chain_step** (**mcmc** *m)

5.21.1 Define Documentation

5.21.1.1 #define MAXIMAL_STEPWIDTH 1000000

Referenced by rmw_adapt_stepwidth().

5.21.1.2 #define MINIMAL_STEPWIDTH 0.0000001

Referenced by rmw_adapt_stepwidth().

5.21.2 Function Documentation

5.21.2.1 unsigned int assess_acceptance_rate (mcmc * *m*, unsigned int *param*, double *desired_acceptance_rate*, double *min_accuracy*, double *max_accuracy*, double * *acceptance_rate*, double * *accuracy*)

Get acceptance rate. The closer the acceptance rate is to the desired acceptance rate, the more accurately will it be assessed.

Parameters

m	
param	
desired_acceptance_rate	
min_accuracy	you can request a upper limit on the accuracy, e.g. 1%. any calculation will have at most the accuracy of 1% then. Otherwise put 0 here.
max_accuracy	you can request a lower limit on the accuracy, e.g. 0.1% any calculation will have at least the accuracy of 0.1% then. Otherwise put 1 here.
acceptance_rate	here the a/r gets stored
accuracy	here the accuracy gets stored

Returns

iterations used

References abs_double, ACCURACY_DEVIATION_FACTOR, assert, clear_bit(), get_bit(), get_n_par(), get_params_accepts_for(), get_params_accepts_global(), IFVERBOSE, markov_chain_step(), markov_chain_step_for(), mcmc_check_best(), reset_accept_rejects(), and set_bit().

Referenced by markov_chain_calibrate_alt(), markov_chain_calibrate_linear_regression(), markov_chain_calibrate_multilinear_regression(), and markov_chain_calibrate_quadratic().

5.21.2.2 void burn_in (mcmc * m, const unsigned int burn_in_iterations)

Perform the given number of burn-in operations

References debug, dump_ul, dump_v, dup_vector(), get_params(), get_steps(), IFVERBOSE, markov_chain_step(), mcmc_check(), mcmc_check_best(), mcmc::params_max, mcmc::params_min, mcmc::params_step, and restart_from_best().

Referenced by calibrate_rest(), and markov_chain_calibrate().

5.21.2.3 void clear_bit (char * bitfield, unsigned int i)

Referenced by assess_acceptance_rate().

5.21.2.4 void do_step_for (mcmc * m, const unsigned int i)

References assert, CIRCULAR_PARAMS, get_next_random_jump(), IFVERBOSE, mod_double, mcmc::params, mcmc::params_max, mcmc::params_min, mcmc::params_step, and set_params_for().

Referenced by markov_chain_step_for().

5.21.2.5 int get_bit (char * bitfield, unsigned int i)

Referenced by assess_acceptance_rate().

5.21.2.6 void markov_chain_step (mcmc * m)

take a step using the markov-chain

Parameters

m	

References calc_model(), dup_vector(), get_prob(), inc_params_accepts(), inc_params_rejects(), mcmc_check(), mcmc::params, and set_params().

Referenced by adapt(), assess_acceptance_rate(), burn_in(), markov_chain_calibrate_orig(), and run_sampler().

5.21.2.7 void markov_chain_step_for (mcmc * m, const unsigned int index)

take a step using the markov-chain for the indexth parameter

Parameters

m	
index	the param to look at

References calc_model_for(), do_step_for(), get_prob(), inc_params_accepts_for(), inc_-params_rejects_for(), mcmc_check(), mcmc::params, and set_params_for().

Referenced by assess_acceptance_rate(), and markov_chain_calibrate_orig().

5.21.2.8 void restart_from_best (mcmc * *m*)

Set the best parameters value found so far as current value, as well as the probability.

References dup_vector(), get_params_best(), get_prob_best(), set_params(), and set_-prob().

Referenced by burn_in(), and markov_chain_calibrate_orig().

5.21.2.9 void rmw_adapt_stepwidth (mcmc * *m,* double *prob_old*)

adapts the step width

References get_n_par(), get_next_uniform_random(), get_prob(), get_steps(), MAXIMAL_-STEPWIDTH, MINIMAL_STEPWIDTH, mcmc::n_iter, mcmc::params_max, mcmc::params_-min, and TARGET_ACCEPTANCE_RATE.

Referenced by adapt().

5.21.2.10 void set_bit (char * *bitfield,* unsigned int *i*)

Referenced by assess_acceptance_rate().

5.22 src/markov_chain.h File Reference

```
#include "mcmc.h"
#include "define_defaults.h"
```

Defines

- #define **DEFAULT_ADJUST_STEP** 0.5
- #define **NO_RESCALING_LIMIT** 15
- #define **ITER_READJUST** 200
- #define **CIRCULAR_PARAMS** 0
- #define **ACCURACY_DEVIATION_FACTOR** 0.25

Functions

- void **markov_chain_calibrate** (**mcmc** *m, const unsigned int burn_in_iterations, double desired_acceptance_rate, const double max_ar_deviation, const unsigned int iter_limit, double mul, const double adjust_step)

- void **markov_chain_step** (**mcmc** ∗m)
- void **markov_chain_step_for** (**mcmc** ∗m, const unsigned int index)
- void **rmw_adapt_stepwidth** (**mcmc** ∗m, double prob_old)
- void **burn_in** (**mcmc** ∗m, const unsigned int burn_in_iterations)
- unsigned int **assess_acceptance_rate** (**mcmc** ∗m, unsigned int param, double desired_acceptance_rate, double min_accuracy, double max_accuracy, double ∗acceptance_rate, double ∗accuracy)
- void **restart_from_best** (**mcmc** ∗m)

5.22.1 Define Documentation

5.22.1.1 #define ACCURACY_DEVIATION_FACTOR 0.25

How good should the acceptance rate be calculated in dependence of deviation from the desired value? accuracy = factor ∗ deviation

Referenced by assess_acceptance_rate().

5.22.1.2 #define CIRCULAR_PARAMS 0

Which parameters are circular?

e.g. if the first and second parameters are angles, you write CIRCULAR_PARAMS=1,2; Do not use zero for the first parameter.

N.B.: if you have a angle, you will want to make it go from 0 to 1 or similar and use pi in your formula. This way it will be more exact.

Referenced by do_step_for().

5.22.1.3 #define DEFAULT_ADJUST_STEP 0.5

Referenced by calibrate_first(), and calibrate_rest().

5.22.1.4 #define ITER_READJUST 200

Referenced by markov_chain_calibrate_orig().

5.22.1.5 #define NO_RESCALING_LIMIT 15

Referenced by markov_chain_calibrate_orig().

5.22.2 Function Documentation

5.22.2.1 unsigned int assess_acceptance_rate (mcmc * *m*, unsigned int *param*, double *desired_acceptance_rate*, double *min_accuracy*, double *max_accuracy*, double * *acceptance_rate*, double * *accuracy*)

Get acceptance rate. The closer the acceptance rate is to the desired acceptance rate, the more accurately will it be assessed.

Parameters

min_accuracy	you can request a upper limit on the accuracy, e.g. 1%. any calculation will have at most the accuracy of 1% then. Otherwise put 0 here.
max_accuracy	you can request a lower limit on the accuracy, e.g. 0.1% any calculation will have at least the accuracy of 0.1% then. Otherwise put 1 here.
acceptance_rate	here the a/r gets stored
accuracy	here the accuracy gets stored

Returns

iterations used

Get acceptance rate. The closer the acceptance rate is to the desired acceptance rate, the more accurately will it be assessed.

Parameters

m	
param	
desired_acceptance_rate	
min_accuracy	you can request a upper limit on the accuracy, e.g. 1%. any calculation will have at most the accuracy of 1% then. Otherwise put 0 here.
max_accuracy	you can request a lower limit on the accuracy, e.g. 0.1% any calculation will have at least the accuracy of 0.1% then. Otherwise put 1 here.
acceptance_rate	here the a/r gets stored
accuracy	here the accuracy gets stored

Returns

iterations used

References abs_double, ACCURACY_DEVIATION_FACTOR, assert, clear_bit(), get_bit(), get_n_par(), get_params_accepts_for(), get_params_accepts_global(), IFVERBOSE, markov_chain_step(), markov_chain_step_for(), mcmc_check_best(), reset_accept_rejects(), and set_bit().

Referenced by markov_chain_calibrate_alt(), markov_chain_calibrate_linear_regression(), markov_chain_calibrate_multilinear_regression(), and markov_chain_calibrate_quadratic().

5.22.2.2 void burn_in (mcmc * *m,* const unsigned int *burn_in_iterations*)

Perform the given number of burn-in operations

References debug, dump_ul, dump_v, dup_vector(), get_params(), get_steps(), IFVERBOSE, markov_chain_step(), mcmc_check(), mcmc_check_best(), mcmc::params_max, mcmc::params_min, mcmc::params_step, and restart_from_best().

Referenced by calibrate_rest(), and markov_chain_calibrate().

5.22.2.3 void markov_chain_calibrate (mcmc * *m,* const unsigned int *burn_in_iterations,* double *desired_acceptance_rate,* const double *max_ar_deviation,* const unsigned int *iter_limit,* double *mul,* const double *adjust_step*)

create/calibrate the markov-chain

Parameters

m	
desired_acceptance_rate	average acceptance rates for individual parameters to be achieved
max_ar_deviation	allowed deviation from desired_acceptance_rate
burn_in_iterations	number of burn-in iterations
iter_limit	number of iterations for step width calibration
mul	factor for adjusting the step width during calibration
adjust_step	gives the factor with which to adjust the stepwidths after burn-in

References burn_in(), dump_d, markov_chain_calibrate_alt(), markov_chain_calibrate_multilinear_regression(), markov_chain_calibrate_orig(), and markov_chain_calibrate_quadratic().

Referenced by calibrate_first(), and calibrate_rest().

5.22.2.4 void markov_chain_step (mcmc * *m*)

take a step using the markov-chain

Parameters

m	

References calc_model(), dup_vector(), get_prob(), inc_params_accepts(), inc_params_rejects(), mcmc_check(), mcmc::params, and set_params().

Referenced by adapt(), assess_acceptance_rate(), burn_in(), markov_chain_calibrate_orig(), and run_sampler().

5.22.2.5 void markov_chain_step_for (mcmc * m, const unsigned int index)

take a step using the markov-chain for the indexth parameter

Parameters

m	
index	the param to look at

References calc_model_for(), do_step_for(), get_prob(), inc_params_accepts_for(), inc_params_rejects_for(), mcmc_check(), mcmc::params, and set_params_for().

Referenced by assess_acceptance_rate(), and markov_chain_calibrate_orig().

5.22.2.6 void restart_from_best (mcmc * m)

Set the best parameters value found so far as current value, as well as the probability.

References dup_vector(), get_params_best(), get_prob_best(), set_params(), and set_prob().

Referenced by burn_in(), and markov_chain_calibrate_orig().

5.22.2.7 void rmw_adapt_stepwidth (mcmc * m, double prob_old)

adapts the step width

References get_n_par(), get_next_uniform_random(), get_prob(), get_steps(), MAXIMAL_STEPWIDTH, MINIMAL_STEPWIDTH, mcmc::n_iter, mcmc::params_max, mcmc::params_min, and TARGET_ACCEPTANCE_RATE.

Referenced by adapt().

5.23 src/markov_chain_calibrate.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_linalg.h>
#include "mcmc.h"
#include "mcmc_internal.h"
#include "debug.h"
#include "gsl_helper.h"
```

Defines

- #define **BETWEEN**(x, min, max) ((x) >= (min) && (x) <= (max))
- #define **MAX**(a, b) ((a) > (b) ? a : b)
- #define **MIN**(a, b) ((a) < (b) ? a : b)
- #define **MAX_ACCURACY_IMPROVEMENT** 2.8
- #define **SCALE_LIN_WORST** 5
- #define **SCALE_MIN** 0.4

Functions

- void **markov_chain_calibrate_multilinear_regression** (**mcmc** *m, double desired_acceptance_rate, const double max_ar_deviation, const unsigned int iter_limit, double mul, const double adjust_step)
- void **markov_chain_calibrate_linear_regression** (**mcmc** *m, double desired_acceptance_rate, const double max_ar_deviation, const unsigned int iter_limit, gsl_matrix *all_stepwidths, gsl_matrix *all_acceptance_rates, gsl_matrix *all_accuracies)
- void **markov_chain_calibrate_quadratic** (**mcmc** *m, double desired_acceptance_rate, const double max_ar_deviation, const unsigned int iter_limit, double mul, const double adjust_step)
- void **markov_chain_calibrate_alt** (**mcmc** *m, double desired_acceptance_rate, const double max_ar_deviation, const unsigned int iter_limit, double mul, const double adjust_step)
- void **markov_chain_calibrate_orig** (**mcmc** *m, double rat_limit, const double max_rat_deviation, const unsigned int iter_limit, double mul, const double adjust_step)
- void **markov_chain_calibrate** (**mcmc** *m, const unsigned int burn_in_iterations, double desired_acceptance_rate, const double max_ar_deviation, const unsigned int iter_limit, double mul, const double adjust_step)

5.23.1 Define Documentation

5.23.1.1 #define BETWEEN(*x, min, max*) ((x) >= (min) && (x) <= (max))

Referenced by markov_chain_calibrate_quadratic().

5.23.1.2 #define MAX(*a, b*) ((a) > (b) ? a : b)

Referenced by markov_chain_calibrate_linear_regression(), markov_chain_calibrate_multilinear_regression(), and markov_chain_calibrate_quadratic().

5.23.1.3 #define MAX_ACCURACY_IMPROVEMENT 2.8

Referenced by markov_chain_calibrate_alt().

5.23.1.4 #define MIN(*a, b*) ((a) < (b) ? a : b)

Referenced by markov_chain_calibrate_quadratic().

5.23.1.5 #define SCALE_LIN_WORST 5

Referenced by markov_chain_calibrate_alt().

5.23.1.6 #define SCALE_MIN 0.4

Referenced by markov_chain_calibrate_alt().

5.23.2 Function Documentation

5.23.2.1 void markov_chain_calibrate (mcmc * *m,* const unsigned int *burn_in_iterations,* double *desired_acceptance_rate,* const double *max_ar_deviation,* const unsigned int *iter_limit,* double *mul,* const double *adjust_step*)

create/calibrate the markov-chain

Parameters

m	
desired_acceptance_rate	average acceptance rates for individual parameters to be achieved
max_ar_deviation	allowed deviation from desired_acceptance_rate
burn_in_iterations	number of burn-in iterations
iter_limit	number of iterations for step width calibration
mul	factor for adjusting the step width during calibration
adjust_step	gives the factor with which to adjust the stepwidths after burn-in

References burn_in(), dump_d, markov_chain_calibrate_alt(), markov_chain_calibrate_multilinear_regression(), markov_chain_calibrate_orig(), and markov_chain_calibrate_quadratic().

Referenced by calibrate_first(), and calibrate_rest().

5.23.2.2 void markov_chain_calibrate_alt (mcmc * *m,* double *desired_acceptance_rate,* const double *max_ar_deviation,* const unsigned int *iter_limit,* double *mul,* const double *adjust_step*)

References abs_double, assert, assess_acceptance_rate(), dump_v, get_n_par(), get_params(), get_steps_for(), get_steps_for_normalized(), MAX_ACCURACY_IMPROVEMENT, SCALE_LIN_WORST, SCALE_MIN, and set_steps_for().

Referenced by markov_chain_calibrate().

5.23.2.3 void markov_chain_calibrate_linear_regression (mcmc * *m,* double *desired_acceptance_rate,* const double *max_ar_deviation,* const unsigned int *iter_limit,* gsl_matrix * *all_stepwidths,* gsl_matrix * *all_acceptance_rates,* gsl_matrix * *all_accuracies*)

References abs_double, assert, assess_acceptance_rate(), debug, get_n_par(), IFDEBUG, MAX, and set_steps_for_normalized().

Referenced by markov_chain_calibrate_quadratic().

5.23.2.4 void markov_chain_calibrate_multilinear_regression (mcmc * *m,* double *desired_acceptance_rate,* const double *max_ar_deviation,* const unsigned int *iter_limit,* double *mul,* const double *adjust_step*)

References abs_double, assess_acceptance_rate(), calc_deviation(), dump_v, get_n_par(), get_random(), get_steps(), get_steps_for_normalized(), IFDEBUG, linreg_n(), MAX, and set_steps_for_normalized().

Referenced by markov_chain_calibrate().

5.23.2.5 void markov_chain_calibrate_orig (mcmc * *m,* double *rat_limit,* const double *max_rat_deviation,* const unsigned int *iter_limit,* double *mul,* const double *adjust_step*)

References abs_double, assert, debug, dump_d, dump_i, dump_ul, dump_v, get_accept_rate(), get_accept_rate_global(), get_n_par(), get_params(), get_params_descr(), get_steps(), get_steps_for(), get_steps_for_normalized(), IFDEBUG, ITER_READJUST, markov_chain_step(), markov_chain_step_for(), mcmc_check_best(), NO_RESCALING_LIMIT, mcmc::params_max, mcmc::params_min, mcmc::params_step, reset_accept_rejects(), restart_from_best(), set_steps_for(), set_steps_for_normalized(), and TARGET_ACCEPTANCE_RATE.

Referenced by markov_chain_calibrate().

5.23.2.6 void markov_chain_calibrate_quadratic (mcmc * *m,* double *desired_acceptance_rate,* const double *max_ar_deviation,* const unsigned int *iter_limit,* double *mul,* const double *adjust_step*)

break condition:

References abs_double, assess_acceptance_rate(), BETWEEN, get_n_par(), get_steps_for_normalized(), IFDEBUG, markov_chain_calibrate_linear_regression(), MAX, max_column(), MIN, min_column(), and set_steps_for_normalized().

Referenced by markov_chain_calibrate().

5.24 src/mcmc.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <libgen.h>
#include "mcmc.h"
#include "gsl_helper.h"
#include "debug.h"
```

Functions

- void **init_seed** (**mcmc** *m)
- **mcmc** * **mcmc_init** (const unsigned int n_pars)
- **mcmc** * **mcmc_free** (**mcmc** *m)
- void **mcmc_check** (const **mcmc** *m)

Variables

- gsl_rng * **r** = NULL

5.24.1 Function Documentation

5.24.1.1 void init_seed (mcmc * m)

References r, and mcmc::random.

Referenced by mcmc_init().

5.24.1.2 void mcmc_check (const mcmc * m)

checks the pointers and dimensions

References assert, mcmc::data, mcmc::n_par, mcmc::params, mcmc::params_best, and mcmc::params_step.

Referenced by burn_in(), calibrate_first(), calibrate_rest(), main(), markov_chain_step(), markov_chain_step_for(), mcmc_load_data(), mcmc_reuse_data(), setup_chains(), and test_write_prob().

5.24.1.3 mcmc* mcmc_free (mcmc * m)

frees the memory used by the class

Returns

NULL for simple assignment `x = mcmc_free(x);`

References mcmc::data, debug, get_n_par(), IFSEGV, mcmc_dump_close(), mem_-free, mcmc::params, mcmc::params_accepts, mcmc::params_best, mcmc::params_descr, mcmc::params_max, mcmc::params_min, mcmc::params_rejects, mcmc::params_step, r, and mcmc::random.

Referenced by prepare_and_run_sampler(), test_append(), test_create(), test_load(), test_random(), test_write(), and test_write_prob().

5.24.1.4 mcmc* mcmc_init (const unsigned int *n_pars*)

Here are the "private" methods of the class and helper functions create class

Parameters

n_pars	parameters

References mcmc::accept, assert, mcmc::data, debug, mcmc::files, IFSEGV, init_seed(), mem_calloc, mem_malloc, mcmc::n_iter, mcmc::n_par, mcmc::params, mcmc::params_-accepts, mcmc::params_best, mcmc::params_descr, mcmc::params_max, mcmc::params_-min, mcmc::params_rejects, mcmc::params_step, mcmc::prior, mcmc::prob, mcmc::prob_-best, and mcmc::reject.

Referenced by mcmc_load_params(), test_append(), and test_create().

5.24.2 Variable Documentation

5.24.2.1 gsl_rng* r = NULL

Referenced by calc_normalized(), countlines(), dup_vector(), get_accept_rate(), init_-seed(), main(), mcmc_dump_close(), mcmc_free(), mcmc_load_params(), and test().

5.25 src/mcmc.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "mcmc_struct.h"
#include "markov_chain.h"
#include "mcmc_gettersetter.h"
```

Defines

- #define **DUMP_FORMAT** "%.15e"
- #define **NOASSERT**
- #define **assert**(cond)

Functions

- **mcmc** ∗ **mcmc_load** (const char ∗filename, const char ∗datafilename)
- **mcmc** ∗ **mcmc_load_params** (const char ∗filename)
- void **mcmc_load_data** (**mcmc** ∗m, const char ∗datafilename)
- void **mcmc_reuse_data** (**mcmc** ∗m, const **mcmc** ∗m_orig)
- **mcmc** ∗ **mcmc_free** (**mcmc** ∗m)
- void **mcmc_check** (const **mcmc** ∗m)
- void **mcmc_append_current_parameters** (**mcmc** ∗m)
- void **mcmc_dump_y_dat** (**mcmc** ∗m, const gsl_vector ∗y_dat, const char ∗filename)
- void **mcmc_dump_flush** (const **mcmc** ∗m)
- void **mcmc_dump_close** (**mcmc** ∗m)
- void **mcmc_open_dump_files** (**mcmc** ∗m, const char ∗suffix, int index, char ∗mode)
- void **mcmc_dump_current** (const **mcmc** ∗m)
- void **mcmc_dump_probabilities** (const **mcmc** ∗m, int n_values, const char ∗suffix)
- void **mcmc_check_best** (**mcmc** ∗m)
- void **calc_model** (**mcmc** ∗m, const gsl_vector ∗old_values)
- void **calc_model_for** (**mcmc** ∗m, const unsigned int i, const double old_value)

5.25.1 Define Documentation

5.25.1.1 #define assert(*cond*)

Referenced by analyse_data_probability(), analyse_marginal_distributions(), assess_-acceptance_rate(), calc_deviation(), calc_marginal_distribution(), calc_model(), calc_-same(), countlines(), do_step_for(), dup_vector(), find_min_max(), get_vector_from_-array(), linreg_n(), main(), markov_chain_calibrate_alt(), markov_chain_calibrate_-linear_regression(), markov_chain_calibrate_orig(), max_vector(), mcmc_check(), mcmc_-dump_close(), mcmc_init(), mcmc_load_params(), mcmc_open_dump_files(), mcmc_-reuse_data(), min_vector(), parallel_tempering_decide_swap_nonrandom(), parallel_-tempering_decide_swap_now(), parallel_tempering_decide_swap_random(), run_sampler(), set_params(), set_params_for(), and setup_chains().

5.25.1.2 #define DUMP_FORMAT "%.15e"

Public methods of the class How many digits should be used for writing numbers out?

If you find the precision too low, increase here. If you find the program too slow, decrease here.

Referenced by calc_marginal_distribution(), main(), mcmc_dump_current(), write_-calibration_summary(), write_calibrations_file(), and write_params_file().

5.25.1.3 #define NOASSERT

Turns additional sanity checks off.

One could think that turning this off would improve performance, but tests have shown the modification is not significant.

5.25.2 Function Documentation

5.25.2.1 void calc_model (mcmc * m, const gsl_vector * old_values)

update the model according to the new parameter values and recalculate the probability for the model

Parameters

m	
old_values	previous values, or NULL

References apply_formula(), assert, mcmc::data, get_beta(), get_n_par(), get_params_for(), get_prior(), Problem::LogLike, p, mcmc::params, Problem::Prior, set_prior(), set_prob(), and SIGMA.

Referenced by calc_model_for(), calibrate_first(), calibrate_rest(), main(), and markov_chain_step().

5.25.2.2 void calc_model_for (mcmc * m, const unsigned int i, const double old_value)

update the model as the new parameter value i changed and recalculate the probability for the model

Parameters

m	
i	index of the parameter value that changed
old_value	previous value of the parameter

References calc_model().

Referenced by main(), and markov_chain_step_for().

5.25.2.3 void mcmc_append_current_parameters (mcmc * m)

adds the current parameter values to params_distr as nth iteration was: add_values

Parameters

m	

References mcmc_dump_current(), and mcmc::n_iter.

Referenced by run_sampler(), test_append(), and test_write_prob().

5.25.2.4 void mcmc_check (const mcmc * m)

checks the pointers and dimensions

References assert, mcmc::data, mcmc::n_par, mcmc::params, mcmc::params_best, and mcmc::params_step.

Referenced by burn_in(), calibrate_first(), calibrate_rest(), main(), markov_chain_step(), markov_chain_step_for(), mcmc_load_data(), mcmc_reuse_data(), setup_chains(), and test_write_prob().

5.25.2.5 void mcmc_check_best (mcmc * m)

check if a new best value has been found

Parameters

m	

References dump_v, mcmc::params, mcmc::prob, mcmc::prob_best, and set_params_best().

Referenced by assess_acceptance_rate(), burn_in(), markov_chain_calibrate_orig(), and run_sampler().

5.25.2.6 void mcmc_dump_close (mcmc * m)

close output files

References assert, mcmc::files, get_n_par(), mem_free, and r.

Referenced by mcmc_free().

5.25.2.7 void mcmc_dump_current (const mcmc * m)

append current parameters to files, unflushed.

References DUMP_FORMAT, mcmc::files, get_n_par(), and mcmc::params.

Referenced by mcmc_append_current_parameters().

5.25.2.8 void mcmc_dump_flush (const mcmc * m)

flush output files

References mcmc::files, and get_n_par().

Referenced by report(), and test_write_prob().

5.25.2.9 void mcmc_dump_probabilities (const mcmc * m, int n_values, const char * suffix)

write probability/distribution (params_distr) out to files.

The files are named after the parameter names, with .prob.dump appended. You can use those files to build a histogram of where the algorithm has been (and how often) to find out

Parameters

m	
n_values	use the n last iterations. if negative, all are used.
suffix	added to output filename as a suffix

5.25.2.10 void mcmc_dump_y_dat (mcmc * m, const gsl_vector * y_dat, const char * filename)

References mcmc::data.

Referenced by test_write().

5.25.2.11 mcmc* mcmc_free (mcmc * m)

frees the memory used by the class

Returns

NULL for simple assignment `x = mcmc_free(x);`

References mcmc::data, debug, get_n_par(), IFSEGV, mcmc_dump_close(), mem_-free, mcmc::params, mcmc::params_accepts, mcmc::params_best, mcmc::params_descr, mcmc::params_max, mcmc::params_min, mcmc::params_rejects, mcmc::params_step, r, and mcmc::random.

Referenced by prepare_and_run_sampler(), test_append(), test_create(), test_load(), test_random(), test_write(), and test_write_prob().

5.25.2.12 mcmc* mcmc_load (const char * filename, const char * datafilename)

create and initialize a mcmc class using the configuration given in

Parameters

filename	file containing the model parameters
datafilename	x/y observed data

Returns

the created mcmc class

References mcmc_load_data(), and mcmc_load_params().

Referenced by test_load(), test_random(), test_write(), and test_write_prob().

5.25.2.13 void mcmc_load_data (mcmc * *m,* const char * *datafilename*)

loads the data from the given file as x/y values

Parameters

m	
datafilename	

References dump_s, IFDEBUGPARSER, and mcmc_check().

Referenced by main(), mcmc_load(), and setup_chains().

5.25.2.14 mcmc* mcmc_load_params (const char * *filename*)

create and initialize a mcmc class using the configuration given in

Parameters

filename	file containing the model parameters

Returns

the created mcmc class

References assert, countlines(), dump_i, IFDEBUGPARSER, mcmc_init(), openfile(), and r.

Referenced by main(), mcmc_load(), and setup_chains().

5.25.2.15 void mcmc_open_dump_files (mcmc * *m,* const char * *suffix,* int *index,* char * *mode*)

open dump files The filenames are created by the scheme paramname+suffix+"-"+index+extension

Parameters

m	
suffix	
index	
mode	'w' for writing, 'a' for append, 'r' for reading

References assert, ASSURE_DUMP_ENABLED, dump_s, mcmc::files, get_n_par(), IFVERBOSE, mem_calloc, mem_free, mcmc::n_par, and mcmc::params_descr.

Referenced by prepare_and_run_sampler(), and test_write_prob().

5.25.2.16 void mcmc_reuse_data (mcmc * m, const mcmc * m_orig)

reference to another object for x and y-data.

Parameters

m	the object to fill
m_orig	the object with loaded data

References assert, mcmc::data, debug, IFDEBUGPARSER, and mcmc_check().

Referenced by setup_chains().

5.26 src/mcmc_calculate.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <libgen.h>
#include "mcmc.h"
#include "mcmc_internal.h"
#include "debug.h"
#include "gsl_helper.h"
```

Functions

- void **mcmc_append_current_parameters** (**mcmc** *m)
- void **mcmc_check_best** (**mcmc** *m)

5.26.1 Function Documentation

5.26.1.1 void mcmc_append_current_parameters (mcmc * m)

adds the current parameter values to params_distr as nth iteration was: add_values

Parameters

m	

References mcmc_dump_current(), and mcmc::n_iter.

Referenced by run_sampler(), test_append(), and test_write_prob().

5.26.1.2 void mcmc_check_best (mcmc * m)

check if a new best value has been found

Parameters

m	

References dump_v, mcmc::params, mcmc::prob, mcmc::prob_best, and set_params_best().

Referenced by assess_acceptance_rate(), burn_in(), markov_chain_calibrate_orig(), and run_sampler().

5.27 src/mcmc_dump.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <libgen.h>
#include "mcmc.h"
#include "debug.h"
```

Defines

- #define **ASSURE_DUMP_ENABLED**

Functions

- void **mcmc_dump_y_dat** (**mcmc** *m, const gsl_vector *y_dat, const char *filename)
- void **mcmc_open_dump_files** (**mcmc** *m, const char *suffix, int index, char *mode)
- void **mcmc_dump_current** (const **mcmc** *m)
- void **mcmc_dump_close** (**mcmc** *m)
- void **mcmc_dump_flush** (const **mcmc** *m)

5.27.1 Define Documentation

5.27.1.1 #define ASSURE_DUMP_ENABLED

Referenced by mcmc_open_dump_files().

5.27.2 Function Documentation

5.27.2.1 void mcmc_dump_close (mcmc * m)

close output files

References assert, mcmc::files, get_n_par(), mem_free, and r.

Referenced by mcmc_free().

5.27.2.2 void mcmc_dump_current (const mcmc * m)

append current parameters to files, unflushed.

References DUMP_FORMAT, mcmc::files, get_n_par(), and mcmc::params.

Referenced by mcmc_append_current_parameters().

5.27.2.3 void mcmc_dump_flush (const mcmc * m)

flush output files

References mcmc::files, and get_n_par().

Referenced by report(), and test_write_prob().

5.27.2.4 void mcmc_dump_y_dat (mcmc * m, const gsl_vector * y_dat, const char * filename)

References mcmc::data.

Referenced by test_write().

5.27.2.5 void mcmc_open_dump_files (mcmc * m, const char * suffix, int index, char * mode)

open dump files The filenames are created by the scheme paramname+suffix+"-"+index+extension

Parameters

m	
suffix	
index	
mode	'w' for writing, 'a' for append, 'r' for reading

References assert, ASSURE_DUMP_ENABLED, dump_s, mcmc::files, get_n_par(), IFVERBOSE, mem_calloc, mem_free, mcmc::n_par, and mcmc::params_descr.

Referenced by prepare_and_run_sampler(), and test_write_prob().

5.28 src/mcmc_gettersetter.c File Reference

```
#include "mcmc.h"
#include "gsl_helper.h"
#include <gsl/gsl_rng.h>
#include <gsl/gsl_randist.h>
#include <gsl/gsl_sf.h>
```

Defines

- #define **N_PARAMETERS**
- #define **get_n_par**(m) N_PARAMETERS
- #define **PROPOSAL**

Functions

- unsigned long **get_params_accepts_sum** (const **mcmc** ∗m)
- unsigned long **get_params_rejects_sum** (const **mcmc** ∗m)
- unsigned long **get_params_accepts_for** (const **mcmc** ∗m, const unsigned int i)
- unsigned long **get_params_rejects_for** (const **mcmc** ∗m, const unsigned int i)
- double **get_accept_rate_for** (const **mcmc** ∗m, const unsigned int i)
- unsigned long **get_params_accepts_global** (const **mcmc** ∗m)
- unsigned long **get_params_rejects_global** (const **mcmc** ∗m)
- double **get_accept_rate_global** (const **mcmc** ∗m)
- gsl_vector ∗ **get_vector_from_array** (const unsigned long ∗array, const unsigned int size)
- gsl_vector ∗ **get_accept_rate** (const **mcmc** ∗m)
- const char ∗∗ **get_params_descr** (const **mcmc** ∗m)
- void **inc_params_accepts_for** (**mcmc** ∗m, const unsigned int i)
- void **inc_params_rejects_for** (**mcmc** ∗m, const unsigned int i)
- void **inc_params_accepts** (**mcmc** ∗m)
- void **inc_params_rejects** (**mcmc** ∗m)
- void **set_params_accepts_for** (**mcmc** ∗m, const long new_params_accept, const unsigned int i)
- void **set_params_rejects_for** (**mcmc** ∗m, const long new_params_reject, const unsigned int i)
- void **reset_accept_rejects** (**mcmc** ∗m)
- void **set_prob_best** (**mcmc** ∗m, const double new_prob_best)
- double **get_prob** (const **mcmc** ∗m)
- double **get_prior** (const **mcmc** ∗m)
- double **get_prob_best** (const **mcmc** ∗m)
- void **set_minmax_for** (**mcmc** ∗m, const double new_min, const double new_max, const unsigned int i)
- void **set_steps_for** (**mcmc** ∗m, const double new_step, const unsigned int i)

- void **set_steps_for_normalized** (**mcmc** ∗m, const double new_step, const unsigned int i)
- void **set_params_best** (**mcmc** ∗m, const gsl_vector ∗new_params_best)
- void **set_params_for** (**mcmc** ∗m, const double new_param, const unsigned int i)
- gsl_vector ∗ **get_params** (const **mcmc** ∗m)
- double **get_params_for** (const **mcmc** ∗m, const unsigned int i)
- gsl_vector ∗ **get_params_min** (const **mcmc** ∗m)
- double **get_params_min_for** (const **mcmc** ∗m, const unsigned int i)
- gsl_vector ∗ **get_params_max** (const **mcmc** ∗m)
- double **get_params_max_for** (const **mcmc** ∗m, const unsigned int i)
- gsl_vector ∗ **get_params_best** (const **mcmc** ∗m)
- double **get_params_best_for** (const **mcmc** ∗m, const unsigned int i)
- void **set_params** (**mcmc** ∗m, gsl_vector ∗new_params)
- void **set_params_descr_all** (**mcmc** ∗m, const char ∗∗new_par_descr)
- void **set_params_descr_for** (**mcmc** ∗m, const char ∗new_par_descr, const unsigned int i)
- gsl_rng ∗ **get_random** (const **mcmc** ∗m)
- void **set_random** (**mcmc** ∗m, gsl_rng ∗newrandom)
- void **set_prob** (**mcmc** ∗m, const double new_prob)
- void **set_prior** (**mcmc** ∗m, const double new_prior)
- const gsl_matrix ∗ **get_data** (const **mcmc** ∗m)
- void **set_data** (**mcmc** ∗m, const gsl_matrix ∗new_data)
- gsl_vector ∗ **get_steps** (const **mcmc** ∗m)
- double **get_steps_for** (const **mcmc** ∗m, const unsigned int i)
- double **get_steps_for_normalized** (const **mcmc** ∗m, const unsigned int i)
- void **free_gsl_vector_array** (gsl_vector ∗∗arr)
- void **set_steps_all** (**mcmc** ∗m, const double ∗new_steps)
- double **get_next_uniform_plusminus_random** (const **mcmc** ∗m)
- double **get_next_uniform_random** (const **mcmc** ∗m)
- double **get_next_random_jump** (const **mcmc** ∗m, const double sigma)
- double **get_next_alog_urandom** (const **mcmc** ∗m)

5.28.1 Define Documentation

5.28.1.1 #define get_n_par(*m*) N_PARAMETERS

Referenced by get_accept_rate(), get_params_accepts_sum(), get_params_rejects_sum(), inc_params_accepts(), inc_params_rejects(), reset_accept_rejects(), and set_steps_all().

5.28.1.2 #define N_PARAMETERS

Fix the number of parameters to the given value

Setting this might leads the compiler to more optimization. If in doubt, do not set, as your program then can be used for any problems regardless of number of parameters.

my favorite trick: automatically count the lines in the current file CCFLAGS="-DN_-PARAMETERS="$(cat params|wc -l)"

Referenced by check().

5.28.1.3 #define PROPOSAL

5.28.2 Function Documentation

5.28.2.1 void free_gsl_vector_array (gsl_vector ∗∗ *arr*)

5.28.2.2 gsl_vector∗ get_accept_rate (const mcmc ∗ *m*)

References get_n_par, get_vector_from_array(), mcmc::params_accepts, mcmc::params_-rejects, and r.

Referenced by markov_chain_calibrate_orig().

5.28.2.3 double get_accept_rate_for (const mcmc ∗ *m*, const unsigned int *i*)

References get_params_accepts_for(), and get_params_rejects_for().

5.28.2.4 double get_accept_rate_global (const mcmc ∗ *m*)

References get_params_accepts_global(), and get_params_rejects_global().

Referenced by markov_chain_calibrate_orig().

5.28.2.5 const gsl_matrix∗ get_data (const mcmc ∗ *m*)

References mcmc::data.

5.28.2.6 double get_next_alog_urandom (const mcmc ∗ *m*)

get next random number (logarithmic uniformly distributed between -inf and 0)

References get_next_uniform_random().

Referenced by test_random().

5.28.2.7 double get_next_random_jump (const mcmc ∗ *m*, const double *sigma*)

get next random number (distributed by the proposal distribution)

You can choose a proposal distribution. The default is a gaussian distribution.

Set PROPOSAL_LOGISTIC, PROPOSAL_UNIFORM if you want to use a different proposal distribution.

References get_random().

Referenced by do_step_for().

5.28.2.8 double get_next_uniform_plusminus_random (const mcmc * m)

get next random number (uniformly distributed between -1 and 1)

References get_next_uniform_random().

5.28.2.9 double get_next_uniform_random (const mcmc * m)

get next random number (uniformly distributed between 0 and 1)

References get_random().

Referenced by get_next_alog_urandom(), get_next_uniform_plusminus_random(), parallel_tempering_decide_swap_now(), parallel_tempering_decide_swap_random(), rmw_adapt_stepwidth(), and test_random().

5.28.2.10 gsl_vector* get_params (const mcmc * m)

References mcmc::params.

Referenced by burn_in(), dump(), main(), markov_chain_calibrate_alt(), markov_chain_calibrate_orig(), print_current_positions(), and read_calibration_file().

5.28.2.11 unsigned long get_params_accepts_for (const mcmc * m, const unsigned int i)

References mcmc::params_accepts.

Referenced by assess_acceptance_rate(), and get_accept_rate_for().

5.28.2.12 unsigned long get_params_accepts_global (const mcmc * m)

References mcmc::accept.

Referenced by assess_acceptance_rate(), dump(), and get_accept_rate_global().

5.28.2.13 unsigned long get_params_accepts_sum (const mcmc * m)

References get_n_par, and mcmc::params_accepts.

Referenced by adapt().

5.28.2.14 gsl_vector* get_params_best (const mcmc * m)

References mcmc::params_best.

Referenced by calibrate_rest(), print_current_positions(), restart_from_best(), and write_-params_file().

5.28.2.15 double get_params_best_for (const mcmc * m, const unsigned int i)

References mcmc::params_best.

5.28.2.16 const char** get_params_descr (const mcmc * m)

References mcmc::params_descr.

Referenced by analyse_marginal_distributions(), calc_marginal_distribution(), markov_-chain_calibrate_orig(), and write_params_file().

5.28.2.17 double get_params_for (const mcmc * m, const unsigned int i)

References mcmc::params.

Referenced by calc_model(), and write_calibrations_file().

5.28.2.18 gsl_vector* get_params_max (const mcmc * m)

References mcmc::params_max.

Referenced by calc_beta_0(), and write_params_file().

5.28.2.19 double get_params_max_for (const mcmc * m, const unsigned int i)

References mcmc::params_max.

Referenced by calc_marginal_distribution(), get_steps_for_normalized(), main(), and set_steps_for_normalized().

5.28.2.20 gsl_vector* get_params_min (const mcmc * m)

References mcmc::params_min.

Referenced by calc_beta_0(), and write_params_file().

5.28.2.21 double get_params_min_for (const mcmc * m, const unsigned int i)

References mcmc::params_min.

Referenced by calc_marginal_distribution(), get_steps_for_normalized(), main(), and set_steps_for_normalized().

5.28.2.22 unsigned long get_params_rejects_for (const mcmc * m, const unsigned int i)

References mcmc::params_rejects.

Referenced by get_accept_rate_for().

5.28.2.23 unsigned long get_params_rejects_global (const mcmc * m)

References mcmc::reject.

Referenced by dump(), and get_accept_rate_global().

5.28.2.24 unsigned long get_params_rejects_sum (const mcmc * m)

References get_n_par, and mcmc::params_rejects.

Referenced by adapt().

5.28.2.25 double get_prior (const mcmc * m)

References mcmc::prior.

Referenced by calc_model(), main(), and run_sampler().

5.28.2.26 double get_prob (const mcmc * m)

References mcmc::prob.

Referenced by adapt(), main(), markov_chain_step(), markov_chain_step_for(), print_current_positions(), rmw_adapt_stepwidth(), and run_sampler().

5.28.2.27 double get_prob_best (const mcmc * m)

References mcmc::prob_best.

Referenced by print_current_positions(), and restart_from_best().

5.28.2.28 gsl_rng* get_random (const mcmc * m)

References mcmc::random.

Referenced by get_next_random_jump(), get_next_uniform_random(), and markov_chain_calibrate_multilinear_regression().

5.28.2.29 gsl_vector* get_steps (const mcmc * m)

References mcmc::params_step.

Referenced by adapt(), burn_in(), calc_beta_0(), calibrate_rest(), markov_chain_calibrate_-multilinear_regression(), markov_chain_calibrate_orig(), rmw_adapt_stepwidth(), write_-calibration_summary(), and write_params_file().

5.28.2.30 double get_steps_for (const mcmc * *m,* const unsigned int *i*)

References mcmc::params_step.

Referenced by get_steps_for_normalized(), markov_chain_calibrate_alt(), markov_-chain_calibrate_orig(), write_calibration_summary(), and write_calibrations_file().

5.28.2.31 double get_steps_for_normalized (const mcmc * *m,* const unsigned int *i*)

References get_params_max_for(), get_params_min_for(), and get_steps_for().

Referenced by markov_chain_calibrate_alt(), markov_chain_calibrate_multilinear_regression(), markov_chain_calibrate_orig(), and markov_chain_calibrate_quadratic().

5.28.2.32 gsl_vector* get_vector_from_array (const unsigned long * *array,* const unsigned int *size*)

References assert.

Referenced by get_accept_rate().

5.28.2.33 void inc_params_accepts (mcmc * *m*)

References mcmc::accept, get_n_par, and inc_params_accepts_for().

Referenced by markov_chain_step().

5.28.2.34 void inc_params_accepts_for (mcmc * *m,* const unsigned int *i*)

References mcmc::params_accepts.

Referenced by inc_params_accepts(), and markov_chain_step_for().

5.28.2.35 void inc_params_rejects (mcmc * *m*)

References get_n_par, inc_params_rejects_for(), and mcmc::reject.

Referenced by markov_chain_step().

5.28.2.36 void inc_params_rejects_for (mcmc * *m,* const unsigned int *i*)

References mcmc::params_rejects.

Referenced by inc_params_rejects(), and markov_chain_step_for().

5.28.2.37 void reset_accept_rejects (mcmc * m)

restart counting

References mcmc::accept, get_n_par, mcmc::reject, set_params_accepts_for(), and set_-params_rejects_for().

Referenced by adapt(), assess_acceptance_rate(), and markov_chain_calibrate_orig().

5.28.2.38 void set_data (mcmc * m, const gsl_matrix * new_data)

References mcmc::data.

Referenced by prepare_and_run_sampler().

5.28.2.39 void set_minmax_for (mcmc * m, const double new_min, const double new_max, const unsigned int i)

References mcmc::params_max, and mcmc::params_min.

5.28.2.40 void set_params (mcmc * m, gsl_vector * new_params)

References assert, mcmc::n_par, and mcmc::params.

Referenced by calibrate_rest(), markov_chain_step(), and restart_from_best().

5.28.2.41 void set_params_accepts_for (mcmc * m, const long new_params_accept, const unsigned int i)

References mcmc::params_accepts.

Referenced by reset_accept_rejects().

5.28.2.42 void set_params_best (mcmc * m, const gsl_vector * new_params_best)

References mcmc::params_best.

Referenced by mcmc_check_best(), and read_calibration_file().

5.28.2.43 void set_params_descr_all (mcmc * m, const char ** new_par_descr)

References mcmc::params_descr.

5.28.2.44 void set_params_descr_for (mcmc * m, const char * new_par_descr, const unsigned int i)

References mcmc::params_descr.

5.28.2.45 void set_params_for (mcmc * m, const double new_param, const unsigned int i)

References assert, and mcmc::params.

Referenced by do_step_for(), main(), markov_chain_step_for(), and read_calibration_file().

5.28.2.46 void set_params_rejects_for (mcmc * m, const long new_params_reject, const unsigned int i)

References mcmc::params_rejects.

Referenced by reset_accept_rejects().

5.28.2.47 void set_prior (mcmc * m, const double new_prior)

References mcmc::prior.

Referenced by calc_model().

5.28.2.48 void set_prob (mcmc * m, const double new_prob)

References mcmc::prob.

Referenced by calc_model(), and restart_from_best().

5.28.2.49 void set_prob_best (mcmc * m, const double new_prob_best)

References mcmc::prob_best.

5.28.2.50 void set_random (mcmc * m, gsl_rng * newrandom)

References mcmc::random.

5.28.2.51 void set_steps_all (mcmc * m, const double * new_steps)

References get_n_par, and set_steps_for().

5.28.2.52 void set_steps_for (mcmc * m, const double new_step, const unsigned int i)

References mcmc::params_step.

Referenced by markov_chain_calibrate_alt(), markov_chain_calibrate_orig(), read_calibration_file(), and set_steps_all().

5.28.2.53 void set_steps_for_normalized (mcmc * *m,* const double *new_step,* const unsigned int *i*)

References get_params_max_for(), get_params_min_for(), and mcmc::params_step.

Referenced by markov_chain_calibrate_linear_regression(), markov_chain_calibrate_-multilinear_regression(), markov_chain_calibrate_orig(), and markov_chain_calibrate_-quadratic().

5.29 src/mcmc_gettersetter.h File Reference

```
#include "mcmc.h"
```

Functions

- const char ** **get_params_descr** (const **mcmc** *m)
- unsigned long **get_params_accepts_global** (const **mcmc** *m)
- unsigned long **get_params_rejects_global** (const **mcmc** *m)
- unsigned long **get_params_accepts_sum** (const **mcmc** *m)
- unsigned long **get_params_rejects_sum** (const **mcmc** *m)
- gsl_vector * **get_accept_rate** (const **mcmc** *m)
- double **get_accept_rate_for** (const **mcmc** *m, const unsigned int i)
- double **get_accept_rate_global** (const **mcmc** *m)
- unsigned long **get_params_accepts_for** (const **mcmc** *m, const unsigned int i)
- unsigned long **get_params_rejects_for** (const **mcmc** *m, const unsigned int i)
- gsl_vector * **get_params** (const **mcmc** *m)
- double **get_params_for** (const **mcmc** *m, const unsigned int i)
- gsl_vector * **get_params_min** (const **mcmc** *m)
- double **get_params_min_for** (const **mcmc** *m, const unsigned int i)
- gsl_vector * **get_params_max** (const **mcmc** *m)
- double **get_params_max_for** (const **mcmc** *m, const unsigned int i)
- gsl_vector * **get_params_best** (const **mcmc** *m)
- double **get_params_best_for** (const **mcmc** *m, const unsigned int i)
- unsigned int **get_n_par** (const **mcmc** *m)
- gsl_rng * **get_random** (const **mcmc** *m)
- double **get_prob** (const **mcmc** *m)
- double **get_prior** (const **mcmc** *m)
- double **get_prob_best** (const **mcmc** *m)
- gsl_vector * **get_steps** (const **mcmc** *m)
- double **get_steps_for** (const **mcmc** *m, const unsigned int i)
- double **get_steps_for_normalized** (const **mcmc** *m, const unsigned int i)
- void **set_prob** (**mcmc** *m, const double new_prob)
- void **set_prob_best** (**mcmc** *m, const double new_prob_best)
- void **set_minmax_for** (**mcmc** *m, const double new_min, const double new_-max, const unsigned int i)
- void **set_model** (**mcmc** *m, gsl_vector *new_model)

- void **set_n_par** (**mcmc** ∗m, const int new_n_par)
- void **set_params_best** (**mcmc** ∗m, const gsl_vector ∗new_params_best)
- void **set_params_for** (**mcmc** ∗m, const double new_param, const unsigned int i)
- void **set_params** (**mcmc** ∗m, gsl_vector ∗new_params)
- void **set_params_descr_all** (**mcmc** ∗m, const char ∗∗new_par_descr)
- void **set_params_descr_for** (**mcmc** ∗m, const char ∗new_par_descr, const unsigned int i)
- void **set_random** (**mcmc** ∗m, gsl_rng ∗newrandom)
- void **set_prior** (**mcmc** ∗m, const double new_prior)
- void **set_data** (**mcmc** ∗m, const gsl_matrix ∗new_data)
- void **set_steps_for** (**mcmc** ∗m, const double new_steps, const unsigned int i)
- void **set_steps_for_normalized** (**mcmc** ∗m, const double new_step, const unsigned int i)
- void **set_steps_all** (**mcmc** ∗m, const double ∗new_steps)
- void **set_params_accepts_for** (**mcmc** ∗m, const long new_params_accept, const unsigned int i)
- void **set_params_rejects_for** (**mcmc** ∗m, const long new_params_reject, const unsigned int i)
- void **inc_params_accepts_for** (**mcmc** ∗m, const unsigned int i)
- void **inc_params_rejects_for** (**mcmc** ∗m, const unsigned int i)
- void **inc_params_accepts** (**mcmc** ∗m)
- void **inc_params_rejects** (**mcmc** ∗m)
- void **reset_accept_rejects** (**mcmc** ∗m)
- double **get_next_uniform_random** (const **mcmc** ∗m)
- double **get_next_uniform_plusminus_random** (const **mcmc** ∗m)
- double **get_next_alog_urandom** (const **mcmc** ∗m)
- double **get_next_random_jump** (const **mcmc** ∗m, const double sigma)

5.29.1 Function Documentation

5.29.1.1 gsl_vector∗ get_accept_rate (const mcmc ∗ *m*)

References get_n_par, get_vector_from_array(), mcmc::params_accepts, mcmc::params_rejects, and r.

Referenced by markov_chain_calibrate_orig().

5.29.1.2 double get_accept_rate_for (const mcmc ∗ *m*, const unsigned int *i*)

References get_params_accepts_for(), and get_params_rejects_for().

5.29.1.3 double get_accept_rate_global (const mcmc ∗ *m*)

References get_params_accepts_global(), and get_params_rejects_global().

Referenced by markov_chain_calibrate_orig().

5.29.1.4 unsigned int get_n_par (const mcmc * m)

Referenced by analyse_marginal_distributions(), assess_acceptance_rate(), calc_model(), calibrate_rest(), dump_mcmc(), main(), markov_chain_calibrate_alt(), markov_chain_calibrate_linear_regression(), markov_chain_calibrate_multilinear_regression(), markov_chain_calibrate_orig(), markov_chain_calibrate_quadratic(), mcmc_dump_close(), mcmc_dump_current(), mcmc_dump_flush(), mcmc_free(), mcmc_open_dump_files(), read_calibration_file(), rmw_adapt_stepwidth(), write_calibration_summary(), write_calibrations_file(), and write_params_file().

5.29.1.5 double get_next_alog_urandom (const mcmc * m)

get next random number (logarithmic uniformly distributed between -inf and 0)

References get_next_uniform_random().

Referenced by test_random().

5.29.1.6 double get_next_random_jump (const mcmc * m, const double sigma)

get next random number (distributed by the proposal distribution)

You can choose a proposal distribution. The default is a gaussian distribution.

Set PROPOSAL_LOGISTIC, PROPOSAL_UNIFORM if you want to use a different proposal distribution.

References get_random().

Referenced by do_step_for().

5.29.1.7 double get_next_uniform_plusminus_random (const mcmc * m)

get next random number (uniformly distributed between -1 and 1)

References get_next_uniform_random().

5.29.1.8 double get_next_uniform_random (const mcmc * m)

get next random number (uniformly distributed between 0 and 1)

References get_random().

Referenced by get_next_alog_urandom(), get_next_uniform_plusminus_random(), parallel_tempering_decide_swap_now(), parallel_tempering_decide_swap_random(), rmw_adapt_stepwidth(), and test_random().

5.29.1.9 gsl_vector* get_params (const mcmc * m)

References mcmc::params.

Referenced by burn_in(), dump(), main(), markov_chain_calibrate_alt(), markov_chain_-calibrate_orig(), print_current_positions(), and read_calibration_file().

5.29.1.10 unsigned long get_params_accepts_for (const mcmc * m, const unsigned int i)

References mcmc::params_accepts.

Referenced by assess_acceptance_rate(), and get_accept_rate_for().

5.29.1.11 unsigned long get_params_accepts_global (const mcmc * m)

References mcmc::accept.

Referenced by assess_acceptance_rate(), dump(), and get_accept_rate_global().

5.29.1.12 unsigned long get_params_accepts_sum (const mcmc * m)

References get_n_par, and mcmc::params_accepts.

Referenced by adapt().

5.29.1.13 gsl_vector* get_params_best (const mcmc * m)

References mcmc::params_best.

Referenced by calibrate_rest(), print_current_positions(), restart_from_best(), and write_-params_file().

5.29.1.14 double get_params_best_for (const mcmc * m, const unsigned int i)

References mcmc::params_best.

5.29.1.15 const char** get_params_descr (const mcmc * m)

References mcmc::params_descr.

Referenced by analyse_marginal_distributions(), calc_marginal_distribution(), markov_-chain_calibrate_orig(), and write_params_file().

5.29.1.16 double get_params_for (const mcmc * m, const unsigned int i)

References mcmc::params.

Referenced by calc_model(), and write_calibrations_file().

5.29.1.17 gsl_vector* get_params_max (const mcmc * m)

References mcmc::params_max.

Referenced by calc_beta_0(), and write_params_file().

5.29.1.18 double get_params_max_for (const mcmc * m, const unsigned int i)

References mcmc::params_max.

Referenced by calc_marginal_distribution(), get_steps_for_normalized(), main(), and set_steps_for_normalized().

5.29.1.19 gsl_vector* get_params_min (const mcmc * m)

References mcmc::params_min.

Referenced by calc_beta_0(), and write_params_file().

5.29.1.20 double get_params_min_for (const mcmc * m, const unsigned int i)

References mcmc::params_min.

Referenced by calc_marginal_distribution(), get_steps_for_normalized(), main(), and set_steps_for_normalized().

5.29.1.21 unsigned long get_params_rejects_for (const mcmc * m, const unsigned int i)

References mcmc::params_rejects.

Referenced by get_accept_rate_for().

5.29.1.22 unsigned long get_params_rejects_global (const mcmc * m)

References mcmc::reject.

Referenced by dump(), and get_accept_rate_global().

5.29.1.23 unsigned long get_params_rejects_sum (const mcmc * m)

References get_n_par, and mcmc::params_rejects.

Referenced by adapt().

5.29.1.24 double get_prior (const mcmc * m)

References mcmc::prior.

Referenced by calc_model(), main(), and run_sampler().

5.29.1.25 double get_prob (const mcmc * m)

References mcmc::prob.

Referenced by adapt(), main(), markov_chain_step(), markov_chain_step_for(), print_current_positions(), rmw_adapt_stepwidth(), and run_sampler().

5.29.1.26 double get_prob_best (const mcmc * m)

References mcmc::prob_best.

Referenced by print_current_positions(), and restart_from_best().

5.29.1.27 gsl_rng* get_random (const mcmc * m)

References mcmc::random.

Referenced by get_next_random_jump(), get_next_uniform_random(), and markov_chain_calibrate_multilinear_regression().

5.29.1.28 gsl_vector* get_steps (const mcmc * m)

References mcmc::params_step.

Referenced by adapt(), burn_in(), calc_beta_0(), calibrate_rest(), markov_chain_calibrate_multilinear_regression(), markov_chain_calibrate_orig(), rmw_adapt_stepwidth(), write_calibration_summary(), and write_params_file().

5.29.1.29 double get_steps_for (const mcmc * m, const unsigned int i)

References mcmc::params_step.

Referenced by get_steps_for_normalized(), markov_chain_calibrate_alt(), markov_chain_calibrate_orig(), write_calibration_summary(), and write_calibrations_file().

5.29.1.30 double get_steps_for_normalized (const mcmc * m, const unsigned int i)

References get_params_max_for(), get_params_min_for(), and get_steps_for().

Referenced by markov_chain_calibrate_alt(), markov_chain_calibrate_multilinear_regression(), markov_chain_calibrate_orig(), and markov_chain_calibrate_quadratic().

5.29.1.31 void inc_params_accepts (mcmc * m)

References mcmc::accept, get_n_par, and inc_params_accepts_for().

Referenced by markov_chain_step().

5.29.1.32 void inc_params_accepts_for (mcmc * m, const unsigned int i)

References mcmc::params_accepts.

Referenced by inc_params_accepts(), and markov_chain_step_for().

5.29.1.33 void inc_params_rejects (mcmc * m)

References get_n_par, inc_params_rejects_for(), and mcmc::reject.

Referenced by markov_chain_step().

5.29.1.34 void inc_params_rejects_for (mcmc * m, const unsigned int i)

References mcmc::params_rejects.

Referenced by inc_params_rejects(), and markov_chain_step_for().

5.29.1.35 void reset_accept_rejects (mcmc * m)

restart counting

References mcmc::accept, get_n_par, mcmc::reject, set_params_accepts_for(), and set_-params_rejects_for().

Referenced by adapt(), assess_acceptance_rate(), and markov_chain_calibrate_orig().

5.29.1.36 void set_data (mcmc * m, const gsl_matrix * new_data)

References mcmc::data.

Referenced by prepare_and_run_sampler().

5.29.1.37 void set_minmax_for (mcmc * m, const double new_min, const double new_max, const unsigned int i)

References mcmc::params_max, and mcmc::params_min.

5.29.1.38 void set_model (mcmc * m, gsl_vector * new_model)

5.29.1.39 void set_n_par (mcmc * m, const int new_n_par)

5.29.1.40 void set_params (mcmc * m, gsl_vector * new_params)

References assert, mcmc::n_par, and mcmc::params.

Referenced by calibrate_rest(), markov_chain_step(), and restart_from_best().

5.29.1.41 void set_params_accepts_for (mcmc * m, const long new_params_accept, const unsigned int i)

References mcmc::params_accepts.

Referenced by reset_accept_rejects().

5.29.1.42 void set_params_best (mcmc * m, const gsl_vector * new_params_best)

References mcmc::params_best.

Referenced by mcmc_check_best(), and read_calibration_file().

5.29.1.43 void set_params_descr_all (mcmc * m, const char ** new_par_descr)

References mcmc::params_descr.

5.29.1.44 void set_params_descr_for (mcmc * m, const char * new_par_descr, const unsigned int i)

References mcmc::params_descr.

5.29.1.45 void set_params_for (mcmc * m, const double new_param, const unsigned int i)

References assert, and mcmc::params.

Referenced by do_step_for(), main(), markov_chain_step_for(), and read_calibration_-file().

5.29.1.46 void set_params_rejects_for (mcmc * m, const long new_params_reject, const unsigned int i)

References mcmc::params_rejects.

Referenced by reset_accept_rejects().

5.29.1.47 void set_prior (mcmc * m, const double new_prior)

References mcmc::prior.

Referenced by calc_model().

5.29.1.48 void set_prob (mcmc * m, const double new_prob)

References mcmc::prob.

Referenced by calc_model(), and restart_from_best().

5.29.1.49 void set_prob_best (mcmc * m, const double new_prob_best)

References mcmc::prob_best.

5.29.1.50 void set_random (mcmc * m, gsl_rng * newrandom)

References mcmc::random.

5.29.1.51 void set_steps_all (mcmc * m, const double * new_steps)

References get_n_par, and set_steps_for().

5.29.1.52 void set_steps_for (mcmc * m, const double new_steps, const unsigned int i)

References mcmc::params_step.

Referenced by markov_chain_calibrate_alt(), markov_chain_calibrate_orig(), read_calibration_file(), and set_steps_all().

5.29.1.53 void set_steps_for_normalized (mcmc * m, const double new_step, const unsigned int i)

References get_params_max_for(), get_params_min_for(), and mcmc::params_step.

Referenced by markov_chain_calibrate_linear_regression(), markov_chain_calibrate_multilinear_regression(), markov_chain_calibrate_orig(), and markov_chain_calibrate_quadratic().

5.30 src/mcmc_internal.h File Reference

```
#include "mcmc.h"
#include <gsl/gsl_histogram.h>
#include <gsl/gsl_sf.h>
```

Defines

- #define **mod_double**(x, div)
- #define **abs_double**(x) ((x) < 0 ? -(x) : (x))

Functions

- unsigned int **countlines** (const char *filename)

5.30.1 Define Documentation

5.30.1.1 #define abs_double(*x*) ((x) < 0 ? -(x) : (x))

the value with positive sign.

Referenced by assess_acceptance_rate(), markov_chain_calibrate_alt(), markov_chain_-calibrate_linear_regression(), markov_chain_calibrate_multilinear_regression(), markov_-chain_calibrate_orig(), and markov_chain_calibrate_quadratic().

5.30.1.2 #define mod_double(*x, div*)

Value:

```
((x) < 0 ? \
        (x) - (div) * (int) ((x) / (div) - 1) : \
        (x) - (div) * (int) ((x) / (div)))
```

a modulo operator for double values

Referenced by do_step_for(), and test_mod().

5.30.2 Function Documentation

5.30.2.1 unsigned int countlines (const char * *filename*)

count the lines (

) in the file

Parameters

filename	

Referenced by mcmc_load_params(), and test_write_prob().

5.31 src/mcmc_parser.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <libgen.h>
#include "mcmc.h"
#include "mcmc_internal.h"
#include "gsl_helper.h"
#include <gsl/gsl_rng.h>
#include "debug.h"
```

```
#include "utils.h"
```

Defines

- #define **MAX_LINE_LENGTH** 256
- #define **IFDEBUGPARSER** if(0)

Functions

- char * **my_strdup** (const char *s)
- void **mcmc_load_data** (**mcmc** *m, const char *datafilename)
- void **mcmc_reuse_data** (**mcmc** *m, const **mcmc** *m_orig)
- **mcmc** * **mcmc_load_params** (const char *filename)
- **mcmc** * **mcmc_load** (const char *filename, const char *datafilename)

5.31.1 Define Documentation

5.31.1.1 #define IFDEBUGPARSER if(0)

Referenced by mcmc_load_data(), mcmc_load_params(), and mcmc_reuse_data().

5.31.1.2 #define MAX_LINE_LENGTH 256

5.31.2 Function Documentation

5.31.2.1 mcmc* mcmc_load (const char * *filename*, const char * *datafilename*)

create and initialize a mcmc class using the configuration given in

Parameters

filename	file containing the model parameters
datafilename	x/y observed data

Returns

the created mcmc class

References mcmc_load_data(), and mcmc_load_params().

Referenced by test_load(), test_random(), test_write(), and test_write_prob().

5.31.2.2 void mcmc_load_data (mcmc * *m*, const char * *datafilename*)

loads the data from the given file as x/y values

Parameters

m	
datafilename	

References dump_s, IFDEBUGPARSER, and mcmc_check().

Referenced by main(), mcmc_load(), and setup_chains().

5.31.2.3 mcmc* mcmc_load_params (const char * *filename*)

create and initialize a mcmc class using the configuration given in

Parameters

filename	file containing the model parameters

Returns

the created mcmc class

References assert, countlines(), dump_i, IFDEBUGPARSER, mcmc_init(), openfile(), and r.

Referenced by main(), mcmc_load(), and setup_chains().

5.31.2.4 void mcmc_reuse_data (mcmc * *m,* const mcmc * *m_orig*)

reference to another object for x and y-data.

Parameters

m	the object to fill
m_orig	the object with loaded data

References assert, mcmc::data, debug, IFDEBUGPARSER, and mcmc_check().

Referenced by setup_chains().

5.31.2.5 char* my_strdup (const char * *s*)

References mem_calloc.

5.32 src/mcmc_struct.h File Reference

```
#include <gsl/gsl_math.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_matrix.h>
```

```
#include <gsl/gsl_rng.h>
```

Data Structures

- struct **mcmc**

5.33 src/memory.h File Reference

```
#include "debug.h"
```

Defines

- #define **FREEMSG**(x) IFSEGV dump_p("about to free", (void∗)x);
- #define **WITHOUT_GARBAGE_COLLECTOR**
- #define **mem_malloc**(x) malloc(x)
- #define **mem_calloc**(n, x) calloc(n, x)
- #define **mem_realloc**(**p**, x) realloc(**p**,x)
- #define **mem_free**(x) { FREEMSG(x); free((void∗)x); }

5.33.1 Define Documentation

5.33.1.1 #define FREEMSG(*x*) IFSEGV dump_p("about to free", (void∗)x);

Enabling the garbage collector

5.33.1.2 #define mem_calloc(*n, x*) calloc(n, x)

Referenced by mcmc_init(), mcmc_open_dump_files(), my_strdup(), run_sampler(), and setup_chains().

5.33.1.3 #define mem_free(*x*) { FREEMSG(x); free((void∗)x); }

Referenced by calibrate_rest(), mcmc_dump_close(), mcmc_free(), mcmc_open_dump_-files(), and prepare_and_run_sampler().

5.33.1.4 #define mem_malloc(*x*) malloc(x)

Referenced by calibrate_rest(), main(), mcmc_init(), and setup_chains().

5.33.1.5 #define mem_realloc(p, x) realloc(p,x)

5.33.1.6 #define WITHOUT_GARBAGE_COLLECTOR

The garbage collector (boehmgc) is enabled by default, but is not required. This disables it.

You may also want to remove the -lgc flag.

5.34 src/parallel_tempering.c File Reference

```
#include <omp.h>
#include "mcmc.h"
#include "parallel_tempering.h"
#include "parallel_tempering_beta.h"
#include "parallel_tempering_interaction.h"
#include "parallel_tempering_config.h"
#include "debug.h"
#include "define_defaults.h"
#include "gsl_helper.h"
#include "parallel_tempering_run.h"
#include "utils.h"
```

Defines

- #define **BURN_IN_ITERATIONS**
- #define **RWM**
- #define **ADAPT**

Functions

- void **register_signal_handlers** ()
- void **run_sampler** (**mcmc** ∗∗chains, int n_beta, unsigned int n_swap, const unsigned long max_iterations, char ∗mode)
- void **report** (const **mcmc** ∗∗chains, const int n_beta)
- void **calibrate_first** ()
- void **calibrate_rest** ()
- void **prepare_and_run_sampler** (const unsigned long max_iterations, int append)
- void **adapt** (**mcmc** ∗∗chains, const unsigned int n_beta, const unsigned int n_swap)

- void **dump** (const **mcmc** **chains, const unsigned int n_beta, const unsigned long iter, FILE *acceptance_file, FILE **probabilities_file)

5.34.1 Define Documentation

5.34.1.1 #define ADAPT

Enable a constant but small rescaling of the step width to keep the acceptance rate up.

5.34.1.2 #define BURN_IN_ITERATIONS

how many iterations should be spent on burn-in

Referenced by calibrate_first(), calibrate_rest(), and check().

5.34.1.3 #define RWM

Enable Random Walk Metropolis (adaptive MCMC method) Also important: **MINIMAL_STEPWIDTH** (p. 44), **MAXIMAL_STEPWIDTH** (p. 44)

5.34.2 Function Documentation

5.34.2.1 void adapt (mcmc ** *chains*, const unsigned int *n_beta*, const unsigned int *n_swap*)

References dump_i, get_params_accepts_sum(), get_params_rejects_sum(), get_prob(), get_steps(), markov_chain_step(), reset_accept_rejects(), rmw_adapt_stepwidth(), and TARGET_ACCEPTANCE_RATE.

Referenced by run_sampler().

5.34.2.2 void calibrate_first ( )

needs: params file BURN_IN_ITERATIONS start values (in params file)

does: calibrate first chain (beta = 1) writes beta, stepwidths and start values as first line in file calibration_result

provides: stepwidths of first chain (calibration_result) new params file (params_suggest) new start values (calibration_result)

References BURN_IN_ITERATIONS, calc_model(), DEFAULT_ADJUST_STEP, ITER_LIMIT, markov_chain_calibrate(), MAX_AR_DEVIATION, mcmc_check(), MUL, setup_chains(), TARGET_ACCEPTANCE_RATE, write_calibrations_file(), and write_params_file().

Referenced by main().

5.34.2.3 void calibrate_rest ( )

needs: params file BURN_IN_ITERATIONS first line in calibration_result BETA_-ALIGNMENT BETA_0 SKIP_CALIBRATE_ALLCHAINS

does: calibrate remaining chains (beta < 1) writes all betas, stepwidths and start values in file calibration_result

provides: stepwidths of first chain (calibration_result) new params file (params_suggest) new start values (calibration_result)

References mcmc::additional_data, BETA_0, burn_in(), BURN_IN_ITERATIONS, calc_-beta_0(), calc_model(), DEFAULT_ADJUST_STEP, dump_vectorln(), dup_vector(), get_beta(), get_chain_beta(), get_n_par(), get_params_best(), get_steps(), ITER_LIMIT, markov_chain_calibrate(), MAX_AR_DEVIATION, mcmc_check(), mem_free, mem_-malloc, MUL, N_BETA, mcmc::params_step, read_calibration_file(), set_beta(), set_-params(), setup_chains(), TARGET_ACCEPTANCE_RATE, write_calibration_summary(), and write_calibrations_file().

Referenced by main().

5.34.2.4 void dump (const mcmc ** *chains*, const unsigned int *n_beta*, const unsigned long *iter*, FILE * *acceptance_file*, FILE ** *probabilities_file*)

References debug, dump_mcmc(), dump_ul, dump_vector(), dumpflag, get_duration(), get_params(), get_params_accepts_global(), get_params_rejects_global(), get_ticks_-per_second(), IFDEBUG, PRINT_PROB_INTERVAL, and report().

Referenced by run_sampler().

5.34.2.5 void prepare_and_run_sampler (const unsigned long *max_iterations*, int *append*)

References mcmc_free(), mcmc_open_dump_files(), mem_free, N_BETA, N_SWAP, read_calibration_file(), register_signal_handlers(), report(), run_sampler(), set_data(), and setup_chains().

Referenced by main().

5.34.2.6 void register_signal_handlers ( )

Referenced by prepare_and_run_sampler().

5.34.2.7 void report (const mcmc ** *chains*, const int *n_beta*)

References mcmc_dump_flush(), and print_current_positions().

Referenced by dump(), and prepare_and_run_sampler().

5.34.2.8 void run_sampler (mcmc ∗∗ *chains,* int *n_beta,* unsigned int *n_swap,* const unsigned long *max_iterations,* char ∗ *mode*)

References adapt(), assert, dump(), dumpflag, get_beta(), get_duration(), get_prior(), get_prob(), markov_chain_step(), mcmc_append_current_parameters(), mcmc_check_best(), mem_calloc, mcmc::n_iter, run, and tempering_interaction().

Referenced by prepare_and_run_sampler().

5.35 src/parallel_tempering.h File Reference

```
#include "mcmc.h"
#include "parallel_tempering_beta.h"
```

Defines

- #define **DUMP_ALL_CHAINS**
- #define **SKIP_CALIBRATE_ALLCHAINS**
- #define **PRINT_PROB_INTERVAL** 1000
- #define **CALIBRATION_FILE** "calibration_results"

Functions

- void **calibrate_first** ()
- void **prepare_and_run_sampler** (unsigned long max_iterations, int append)
- void **calibrate_rest** ()
- void **analyse_marginal_distributions** ()
- void **analyse_data_probability** ()

5.35.1 Define Documentation

5.35.1.1 #define CALIBRATION_FILE "calibration_results"

Referenced by help_phase(), read_calibration_file(), and write_calibrations_file().

5.35.1.2 #define DUMP_ALL_CHAINS

should all chains be dumped?

Otherwise, only chain0 is dumped (beta = 1)

5.35.1.3 #define PRINT_PROB_INTERVAL 1000

After how many iterations do you want the program to write out information?

Note that this should be a multiple of N_SWAP, otherwise you get weird effects, since this is checked after N_SWAP iterations using `(iter % PRINT_PROB_INTERVAL == 0)`.

Referenced by check(), and dump().

5.35.1.4 #define SKIP_CALIBRATE_ALLCHAINS

Enabling this will only calibrate the first two chains, not all of them. The stepwidths for the rest of the chains will be predicted.

5.35.2 Function Documentation

5.35.2.1 void analyse_data_probability ( )

References assert, dump_s, get_beta(), N_BETA, read_calibration_file(), and setup_chains().

Referenced by main().

5.35.2.2 void analyse_marginal_distributions ( )

References assert, calc_marginal_distribution(), get_n_par(), get_params_descr(), GNUPLOT_STYLE, N_BETA, read_calibration_file(), and setup_chains().

Referenced by main().

5.35.2.3 void calibrate_first ( )

applications can run the follwing functions

needs: params file BURN_IN_ITERATIONS start values (in params file)

does: calibrate first chain (beta = 1) writes beta, stepwidths and start values as first line in file calibration_result

provides: stepwidths of first chain (calibration_result) new params file (params_suggest) new start values (calibration_result)

References BURN_IN_ITERATIONS, calc_model(), DEFAULT_ADJUST_STEP, ITER_LIMIT, markov_chain_calibrate(), MAX_AR_DEVIATION, mcmc_check(), MUL, setup_chains(), TARGET_ACCEPTANCE_RATE, write_calibrations_file(), and write_params_file().

Referenced by main().

5.35.2.4 void calibrate_rest ( )

needs: params file BURN_IN_ITERATIONS first line in calibration_result BETA_-ALIGNMENT BETA_0 SKIP_CALIBRATE_ALLCHAINS

does: calibrate remaining chains (beta < 1) writes all betas, stepwidths and start values in file calibration_result

provides: stepwidths of first chain (calibration_result) new params file (params_suggest) new start values (calibration_result)

References mcmc::additional_data, BETA_0, burn_in(), BURN_IN_ITERATIONS, calc_-beta_0(), calc_model(), DEFAULT_ADJUST_STEP, dump_vectorln(), dup_vector(), get_beta(), get_chain_beta(), get_n_par(), get_params_best(), get_steps(), ITER_LIMIT, markov_chain_calibrate(), MAX_AR_DEVIATION, mcmc_check(), mem_free, mem_-malloc, MUL, N_BETA, mcmc::params_step, read_calibration_file(), set_beta(), set_-params(), setup_chains(), TARGET_ACCEPTANCE_RATE, write_calibration_summary(), and write_calibrations_file().

Referenced by main().

5.35.2.5 void prepare_and_run_sampler (unsigned long *max_iterations,* int *append*)

References mcmc_free(), mcmc_open_dump_files(), mem_free, N_BETA, N_SWAP, read_calibration_file(), register_signal_handlers(), report(), run_sampler(), set_data(), and setup_chains().

Referenced by main().

5.36 src/parallel_tempering_beta.c File Reference

```
#include "mcmc.h"
#include "gsl_helper.h"
#include "debug.h"
#include "parallel_tempering_beta.h"
#include "define_defaults.h"
```

Functions

- void **set_beta** (**mcmc** *m, double newbeta)
- double **get_beta** (const **mcmc** *m)
- void **inc_swapcount** (**mcmc** *m)
- unsigned long **get_swapcount** (const **mcmc** *m)
- void **print_current_positions** (const **mcmc** **chains, const int n_beta)
- double **equidistant_beta** (const unsigned int i, const unsigned int n_beta, const double beta_0)

- double **equidistant_temperature** (const unsigned int i, const unsigned int n_beta, const double beta_0)
- double **chebyshev_temperature** (const unsigned int i, const unsigned int n_beta, const double beta_0)
- double **chebyshev_beta** (const unsigned int i, const unsigned int n_beta, const double beta_0)
- double **equidistant_stepwidth** (const unsigned int i, const unsigned int n_beta, const double beta_0)
- double **chebyshev_stepwidth** (const unsigned int i, const unsigned int n_beta, const double beta_0)
- double **hot_chains** (const unsigned int i, const unsigned int n_beta, const double beta_0)
- double **get_chain_beta** (unsigned int i, unsigned int n_beta, double beta_0)
- double **calc_beta_0** (**mcmc** *m, gsl_vector *stepwidth_factors)

5.36.1 Function Documentation

5.36.1.1 double calc_beta_0 (mcmc * *m,* gsl_vector * *stepwidth_factors*)

References BETA_0_STEPWIDTH, dup_vector(), get_params_max(), get_params_min(), and get_steps().

Referenced by calibrate_rest().

5.36.1.2 double chebyshev_beta (const unsigned int *i,* const unsigned int *n_beta,* const double *beta_0*)

5.36.1.3 double chebyshev_stepwidth (const unsigned int *i,* const unsigned int *n_beta,* const double *beta_0*)

5.36.1.4 double chebyshev_temperature (const unsigned int *i,* const unsigned int *n_beta,* const double *beta_0*)

5.36.1.5 double equidistant_beta (const unsigned int *i,* const unsigned int *n_beta,* const double *beta_0*)

5.36.1.6 double equidistant_stepwidth (const unsigned int *i,* const unsigned int *n_beta,* const double *beta_0*)

5.36.1.7 double equidistant_temperature (const unsigned int *i,* const unsigned int *n_beta,* const double *beta_0*)

5.36.1.8 double get_beta (const mcmc * *m*)

References mcmc::additional_data.

Referenced by analyse_data_probability(), calc_model(), calibrate_rest(), run_sampler(), write_calibration_summary(), and write_calibrations_file().

5.36.1.9 double get_chain_beta (unsigned int *i,* unsigned int *n_beta,* double *beta_0*)

References BETA_ALIGNMENT.

Referenced by calibrate_rest(), and write_calibration_summary().

5.36.1.10 unsigned long get_swapcount (const mcmc * *m*)

References mcmc::additional_data.

Referenced by print_current_positions().

5.36.1.11 double hot_chains (const unsigned int *i,* const unsigned int *n_beta,* const double *beta_0*)

5.36.1.12 void inc_swapcount (mcmc * *m*)

References mcmc::additional_data.

Referenced by tempering_interaction().

5.36.1.13 void print_current_positions (const mcmc ** *chains,* const int *n_beta*)

References dump_vectorln(), get_params(), get_params_best(), get_prob(), get_prob_-best(), and get_swapcount().

Referenced by report().

5.36.1.14 void set_beta (mcmc * *m,* double *newbeta*)

References mcmc::additional_data.

Referenced by calibrate_rest(), main(), read_calibration_file(), and setup_chains().

5.37 src/parallel_tempering_beta.h File Reference

```
#include <gsl/gsl_sf.h>
#include "mcmc.h"
#include "parallel_tempering.h"
#include "parallel_tempering_interaction.h"
```

Data Structures

- struct **parallel_tempering_mcmc**

Defines

- #define **BETA_ALIGNMENT**
- #define **BETA_0_STEPWIDTH** 1.0

Functions

- void **set_beta** (**mcmc** ∗m, double newbeta)
- double **get_beta** (const **mcmc** ∗m)
- void **inc_swapcount** (**mcmc** ∗m)
- unsigned long **get_swapcount** (const **mcmc** ∗m)
- void **print_current_positions** (const **mcmc** ∗∗chains, const int n_beta)
- double **get_chain_beta** (unsigned int i, unsigned int n_beta, double beta_0)
- double **calc_beta_0** (**mcmc** ∗m, gsl_vector ∗stepwidth_factors)

5.37.1 Define Documentation

5.37.1.1 #define BETA_0_STEPWIDTH 1.0

hottest chain stepwidth in units of parameter space

Referenced by calc_beta_0().

5.37.1.2 #define BETA_ALIGNMENT

Defines how the beta value should be assigned/distributed between the chains.

beta = 1 / temperature.

You can choose equidistant (linear) alignment of the temperature or beta. Or, what often proves to be a good choice, you can use Chebyshev nodes for the values of the temperature or beta.

e.g.: BETA_ALIGNMENT=equidistant_beta

also available: equidistant_temperature, chebyshev_beta, chebyshev_temperature, equidistant_-stepwidth, chebyshev_stepwidth

default: chebyshev_beta

Referenced by check(), and get_chain_beta().

5.37.2 Function Documentation

5.37.2.1 double calc_beta_0 (mcmc ∗ *m,* gsl_vector ∗ *stepwidth_factors*)

References BETA_0_STEPWIDTH, dup_vector(), get_params_max(), get_params_-min(), and get_steps().

Referenced by calibrate_rest().

5.37.2.2 double get_beta (const mcmc * m)

References mcmc::additional_data.

Referenced by analyse_data_probability(), calc_model(), calibrate_rest(), run_sampler(), write_calibration_summary(), and write_calibrations_file().

5.37.2.3 double get_chain_beta (unsigned int *i*, unsigned int *n_beta*, double *beta_0*)

References BETA_ALIGNMENT.

Referenced by calibrate_rest(), and write_calibration_summary().

5.37.2.4 unsigned long get_swapcount (const mcmc * m)

References mcmc::additional_data.

Referenced by print_current_positions().

5.37.2.5 void inc_swapcount (mcmc * m)

References mcmc::additional_data.

Referenced by tempering_interaction().

5.37.2.6 void print_current_positions (const mcmc ** *chains*, const int *n_beta*)

References dump_vectorln(), get_params(), get_params_best(), get_prob(), get_prob_-best(), and get_swapcount().

Referenced by report().

5.37.2.7 void set_beta (mcmc * m, double *newbeta*)

References mcmc::additional_data.

Referenced by calibrate_rest(), main(), read_calibration_file(), and setup_chains().

5.38 src/parallel_tempering_config.c File Reference

```
#include <omp.h>
#include "mcmc.h"
#include "parallel_tempering_config.h"
#include "debug.h"
#include "define_defaults.h"
```

```
#include "gsl_helper.h"
#include "utils.h"
```

Functions

- void **write_params_file** (**mcmc** ∗m)
- void **write_calibration_summary** (**mcmc** ∗∗chains, unsigned int n_chains)
- **mcmc** ∗∗ **setup_chains** ()
- void **read_calibration_file** (**mcmc** ∗∗chains, unsigned int n_chains)
- void **write_calibrations_file** (**mcmc** ∗∗chains, const unsigned int n_chains)

5.38.1 Function Documentation

5.38.1.1 void read_calibration_file (mcmc ∗∗ *chains,* unsigned int *n_chains*)

read betas, stepwidths and start values of all chains

Returns

lines read

References CALIBRATION_FILE, get_n_par(), get_params(), set_beta(), set_params_best(), set_params_for(), and set_steps_for().

Referenced by analyse_data_probability(), analyse_marginal_distributions(), calibrate_rest(), and prepare_and_run_sampler().

5.38.1.2 mcmc∗∗ setup_chains ( )

References mcmc::additional_data, assert, DATA_FILENAME, mcmc_check(), mcmc_load_data(), mcmc_load_params(), mcmc_reuse_data(), mem_calloc, mem_malloc, N_BETA, PARAMS_FILENAME, and set_beta().

Referenced by analyse_data_probability(), analyse_marginal_distributions(), calibrate_first(), calibrate_rest(), and prepare_and_run_sampler().

5.38.1.3 void write_calibration_summary (mcmc ∗∗ *chains,* unsigned int *n_chains*)

References DUMP_FORMAT, dup_vector(), get_beta(), get_chain_beta(), get_n_par(), get_steps(), and get_steps_for().

Referenced by calibrate_rest().

5.38.1.4 void write_calibrations_file (mcmc ∗∗ *chains,* const unsigned int *n_chains*)

References CALIBRATION_FILE, DUMP_FORMAT, get_beta(), get_n_par(), get_params_for(), and get_steps_for().

Referenced by calibrate_first(), and calibrate_rest().

5.38.1.5 void write_params_file (mcmc * m)

References DUMP_FORMAT, get_n_par(), get_params_best(), get_params_descr(), get_params_max(), get_params_min(), get_steps(), and PARAMS_FILENAME.

Referenced by calibrate_first().

5.39 src/parallel_tempering_config.h File Reference

```
#include "mcmc.h"
#include "parallel_tempering_beta.h"
```

Defines

- #define **CALIBRATION_FILE** "calibration_results"

Functions

- void **write_params_file** (**mcmc** *m)
- void **write_calibration_summary** (**mcmc** **chains, unsigned int n_chains)
- **mcmc** ** **setup_chains** ()
- void **read_calibration_file** (**mcmc** **chains, unsigned int n_chains)
- void **write_calibrations_file** (**mcmc** **chains, const unsigned int n_chains)

5.39.1 Define Documentation

5.39.1.1 #define CALIBRATION_FILE "calibration_results"

5.39.2 Function Documentation

5.39.2.1 void read_calibration_file (mcmc ** chains, unsigned int n_chains)

read betas, stepwidths and start values of all chains

Returns

lines read

References CALIBRATION_FILE, get_n_par(), get_params(), set_beta(), set_params_best(), set_params_for(), and set_steps_for().

Referenced by analyse_data_probability(), analyse_marginal_distributions(), calibrate_rest(), and prepare_and_run_sampler().

5.39.2.2 mcmc** setup_chains ( )

References mcmc::additional_data, assert, DATA_FILENAME, mcmc_check(), mcmc_load_data(), mcmc_load_params(), mcmc_reuse_data(), mem_calloc, mem_malloc, N_BETA, PARAMS_FILENAME, and set_beta().

Referenced by analyse_data_probability(), analyse_marginal_distributions(), calibrate_first(), calibrate_rest(), and prepare_and_run_sampler().

5.39.2.3 void write_calibration_summary (mcmc ** *chains*, unsigned int *n_chains*)

References DUMP_FORMAT, dup_vector(), get_beta(), get_chain_beta(), get_n_par(), get_steps(), and get_steps_for().

Referenced by calibrate_rest().

5.39.2.4 void write_calibrations_file (mcmc ** *chains*, const unsigned int *n_chains*)

References CALIBRATION_FILE, DUMP_FORMAT, get_beta(), get_n_par(), get_params_for(), and get_steps_for().

Referenced by calibrate_first(), and calibrate_rest().

5.39.2.5 void write_params_file (mcmc * *m*)

References DUMP_FORMAT, get_n_par(), get_params_best(), get_params_descr(), get_params_max(), get_params_min(), get_steps(), and PARAMS_FILENAME.

Referenced by calibrate_first().

5.40 src/parallel_tempering_interaction.c File Reference

```
#include "parallel_tempering_interaction.h"
#include "parallel_tempering.h"
#include "debug.h"
#include "mcmc_internal.h"
#include "gsl_helper.h"
```

Functions

- int **parallel_tempering_decide_swap_random** (**mcmc** **chains, int n_beta, int n_swap)
- int **parallel_tempering_decide_swap_nonrandom** (**mcmc** **chains, int n_beta, int n_swap, int iter)
- int **parallel_tempering_decide_swap_now** (**mcmc** **chains, int n_beta)

- void **tempering_interaction** (**mcmc** ∗∗chains, unsigned int n_beta, unsigned long iter)

5.40.1 Function Documentation

5.40.1.1 int parallel_tempering_decide_swap_nonrandom (mcmc ∗∗ *chains,* int *n_beta,* int *n_swap,* int *iter*)

chooses one chain after another to swap. Swaps occur every n_swap.

References assert.

5.40.1.2 int parallel_tempering_decide_swap_now (mcmc ∗∗ *chains,* int *n_beta*)

chooses one chain to swap by random. Swap occurs every time.

References assert, and get_next_uniform_random().

Referenced by tempering_interaction().

5.40.1.3 int parallel_tempering_decide_swap_random (mcmc ∗∗ *chains,* int *n_beta,* int *n_swap*)

chooses a chain to swap by random. Swaps occur with probability 1/n_swap

References assert, debug, get_next_uniform_random(), and IFVERBOSE.

Referenced by tempering_interaction().

5.40.1.4 void tempering_interaction (mcmc ∗∗ *chains,* unsigned int *n_beta,* unsigned long *iter*)

does swapping chains and other mixes.

Parameters

chains	
iter	number of iterations that passed. Is a multiple of n_swap.
n_beta	size of chains

References inc_swapcount(), parallel_tempering_decide_swap_now(), and parallel_-tempering_decide_swap_random().

Referenced by run_sampler().

5.41 src/parallel_tempering_interaction.h File Reference

```
#include "mcmc.h"
```

Defines

- #define **RANDOMSWAP**

Functions

- void **tempering_interaction** (**mcmc** **chains, unsigned int n_beta, unsigned long iter)

5.41.1 Define Documentation

5.41.1.1 #define RANDOMSWAP

this shouldn't make any difference at the moment ...

5.41.2 Function Documentation

5.41.2.1 void tempering_interaction (mcmc ** *chains*, unsigned int *n_beta*, unsigned long *iter*)

does swapping chains and other mixes.

Parameters

chains	
iter	number of iterations that passed. Is a multiple of n_swap.
n_beta	size of chains

References inc_swapcount(), parallel_tempering_decide_swap_now(), and parallel_tempering_decide_swap_random().

Referenced by run_sampler().

5.42 src/parallel_tempering_run.c File Reference

```
#include "mcmc.h"
#include "parallel_tempering_run.h"
#include <signal.h>
#include <time.h>
```

Functions

- void **ctrl_c_handler** (int signalnr)
- void **sigusr_handler** (int signalnr)

- int **get_duration** ()
- void **register_signal_handlers** ()
- long unsigned int **get_ticks_per_second** ()

Variables

- int **run** = 1
- int **dumpflag**

5.42.1 Function Documentation

5.42.1.1 void ctrl_c_handler (int *signalnr*)

References run.

Referenced by register_signal_handlers().

5.42.1.2 int get_duration ( )

Referenced by dump(), and run_sampler().

5.42.1.3 long unsigned int get_ticks_per_second ( )

Referenced by dump().

5.42.1.4 void register_signal_handlers ( )

References ctrl_c_handler(), and sigusr_handler().

Referenced by prepare_and_run_sampler().

5.42.1.5 void sigusr_handler (int *signalnr*)

References dumpflag.

Referenced by register_signal_handlers().

5.42.2 Variable Documentation

5.42.2.1 int dumpflag

Referenced by dump(), run_sampler(), and sigusr_handler().

5.42.2.2 int run = 1

Referenced by ctrl_c_handler(), and run_sampler().

5.43 src/parallel_tempering_run.h File Reference

Functions

- int **get_duration** ()
- void **register_signal_handlers** ()
- long unsigned int **get_ticks_per_second** ()

Variables

- int **run**
- int **dumpflag**

5.43.1 Function Documentation

5.43.1.1 int get_duration ( )

Referenced by dump(), and run_sampler().

5.43.1.2 long unsigned int get_ticks_per_second ( )

Referenced by dump().

5.43.1.3 void register_signal_handlers ( )

References ctrl_c_handler(), and sigusr_handler().

5.43.2 Variable Documentation

5.43.2.1 int dumpflag

Referenced by dump(), run_sampler(), and sigusr_handler().

5.43.2.2 int run

Referenced by ctrl_c_handler(), and run_sampler().

5.44 src/utils.c File Reference

```
#include "utils.h"
#include <ctype.h>
```

Functions

- FILE * **openfile** (const char *filename)
- unsigned int **countlines** (const char *filename)
- unsigned int **get_column_count** (const char *filename)

5.44.1 Function Documentation

5.44.1.1 unsigned int countlines (const char * *filename*)

count the lines (

) in the file

Parameters

filename	

References assert, openfile(), and r.

Referenced by mcmc_load_params(), and test_write_prob().

5.44.1.2 unsigned int get_column_count (const char * *filename*)

how many columns does this file have? (looks at first line only)

References openfile().

Referenced by calc_marginal_distribution().

5.44.1.3 FILE* openfile (const char * *filename*)

open the file or die

Referenced by append_to_hists(), calc_mcmc_error(), countlines(), find_min_max(), get_column_count(), and mcmc_load_params().

5.45 src/utils.h File Reference

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

```
#include <assert.h>
```

Functions

- FILE * **openfile** (const char *filename)
- unsigned int **countlines** (const char *filename)
- unsigned int **get_column_count** (const char *filename)

5.45.1 Function Documentation

5.45.1.1 unsigned int countlines (const char * *filename*)

a wc -l on the file.

count the lines (

) in the file

Parameters

filename	

References assert, openfile(), and r.

5.45.1.2 unsigned int get_column_count (const char * *filename*)

how many columns does this file have? (looks at first line only)

References openfile().

Referenced by calc_marginal_distribution().

5.45.1.3 FILE* openfile (const char * *filename*)

open the file or die

Referenced by append_to_hists(), calc_mcmc_error(), countlines(), find_min_max(), get_column_count(), and mcmc_load_params().

5.46 tests/run-tests.c File Reference

```
#include <ctype.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
```

Functions

- int **count_tests** ()
- int **test** (int testnr)
- int **test_all** ()
- void **usage** (char ∗**progname**)
- int **main** (int argc, char ∗∗argv)

Variables

- int(∗ **tests_registration** [ ])(void)

5.46.1 Function Documentation

5.46.1.1 int count_tests ( )

References tests_registration.

Referenced by test(), test_all(), test_tests(), and usage().

5.46.1.2 int main (int *argc,* char ∗∗ *argv*)

References test(), test_all(), and usage().

5.46.1.3 int test (int *testnr*)

References count_tests(), r, and tests_registration.

Referenced by main(), and test_all().

5.46.1.4 int test_all ( )

References count_tests(), and test().

Referenced by main().

5.46.1.5 void usage (char ∗ *progname*)

References count_tests().

5.46.2 Variable Documentation

5.46.2.1 int(∗ tests_registration[ ])(void)

Referenced by count_tests(), and test().

5.47 tests/tests.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "mcmc.h"
#include "mcmc_internal.h"
#include "debug.h"
#include "gsl_helper.h"
#include "histogram.h"
```

Defines

- #define **CIRCULAR_PARAMS** 1,2
- #define **DUMPONFAIL** 1
- #define **ASSERTDUMP**(condition, text, a)
- #define **ASSERT**(condition, text) ASSERTDUMP(condition, text, NULL)
- #define **ASSERTEQUALI**(result, expected, text)
- #define **ASSERTEQUALD**(result, expected, text)

Functions

- int **count_tests** ()
- int **test_tests** (void)
- int **test_hist** (void)
- int **test_create** (void)
- int **test_load** (void)
- int **test_append** (void)
- int **test_random** (void)
- int **test_mod** (void)
- int **test_write** (void)
- int **test_write_prob** (void)
- void **calc_prob** (**mcmc** ∗m)
- void **calc_model** (**mcmc** ∗m, const gsl_vector ∗old_values)
- void **calc_model_for** (**mcmc** ∗m, const unsigned int index, const double old_-value)

Variables

- int(∗ **tests_registration** [ ])(void)

5.47.1 Define Documentation

5.47.1.1 #define ASSERT(*condition, text*) ASSERTDUMP(condition, text, NULL)

Referenced by test_load(), and test_random().

5.47.1.2 #define ASSERTDUMP(*condition, text, a*)

Value:

```
{ \
                if(!(condition)) { \
                        printf("  ASSERT FAILED: %s\n",text); \
                        if(DUMPONFAIL && a != NULL){ \
                                printf("    dump: \n"); \
                                dump_mcmc(a); \
                        } \
                        return 1; \
                } else { \
                        printf("  subtest ok: %s\n",text); \
                } \
        }
```

5.47.1.3 #define ASSERTEQUALD(*result, expected, text*)

Value:

```
{ \
                if((expected) == (result) || \
                        abs_double((expected) - (result)) < abs_double((expected)
      +(result))*0.001 \
                ) { \
                        printf("  subtest ok: %s\n", text); \
                } else { \
                        printf("  ASSERT EQUAL FAILED: expected: %e; got: %e; %s\
      n", expected, result, text); \
                        printf("  ASSERT CODE TERM: %s\n", #result); \
                        return 1; \
                } \
          }
```

Referenced by test_hist(), test_load(), test_mod(), and test_random().

5.47.1.4 #define ASSERTEQUALI(*result, expected, text*)

Value:

```
{ \
                if(expected != result) { \
                        printf("  ASSERT EQUAL FAILED: expected: %d; got: %d; %s\
      n", expected, result, text); \
                        printf("  ASSERT CODE: %s\n", #expected); \
                        return 1; \
                } else { \
```

```
                printf("  subtest ok: %s\n", text); \
        } \
}
```

Referenced by test_append(), test_create(), test_hist(), test_load(), and test_write_-prob().

5.47.1.5 #define CIRCULAR_PARAMS 1,2

5.47.1.6 #define DUMPONFAIL 1

5.47.2 Function Documentation

5.47.2.1 void calc_model (mcmc * *m,* const gsl_vector * *old_values*)

update the model according to the new parameter values and recalculate the probability for the model

Parameters

m	
old_values	previous values, or NULL

Referenced by calc_model_for(), calibrate_first(), calibrate_rest(), main(), and markov_-chain_step().

5.47.2.2 void calc_model_for (mcmc * *m,* const unsigned int *i,* const double *old_value*)

update the model as the new parameter value i changed and recalculate the probability for the model

Parameters

m	
i	index of the parameter value that changed
old_value	previous value of the parameter

Referenced by main(), and markov_chain_step_for().

5.47.2.3 void calc_prob (mcmc * *m*)

5.47.2.4 int count_tests ( )

References tests_registration.

Referenced by test(), test_all(), test_tests(), and usage().

5.47.2.5 int test_append (void)

References ASSERTEQUALI, mcmc_append_current_parameters(), mcmc_free(), mcmc_-init(), and mcmc::n_iter.

5.47.2.6 int test_create (void)

References ASSERTEQUALI, debug, dump_mcmc(), mcmc_free(), mcmc_init(), and mcmc::n_par.

5.47.2.7 int test_hist (void)

References ASSERTEQUALD, ASSERTEQUALI, and calc_hist().

5.47.2.8 int test_load (void)

References ASSERT, ASSERTEQUALD, ASSERTEQUALI, mcmc::data, mcmc_free(), mcmc_load(), mcmc::n_par, mcmc::params, mcmc::params_descr, mcmc::params_-max, mcmc::params_min, and mcmc::params_step.

5.47.2.9 int test_mod (void)

References ASSERTEQUALD, and mod_double.

5.47.2.10 int test_random (void)

References ASSERT, ASSERTEQUALD, get_next_alog_urandom(), get_next_uniform_-random(), mcmc_free(), and mcmc_load().

5.47.2.11 int test_tests (void)

References count_tests().

5.47.2.12 int test_write (void)

References mcmc::data, debug, mcmc_dump_y_dat(), mcmc_free(), and mcmc_load().

5.47.2.13 int test_write_prob (void)

References ASSERTEQUALI, countlines(), debug, mcmc_append_current_parameters(), mcmc_check(), mcmc_dump_flush(), mcmc_free(), mcmc_load(), mcmc_open_dump_-files(), mcmc::params, mcmc::params_max, and require.

5.47.3 Variable Documentation

5.47.3.1 int(* tests_registration[])(void)

Initial value:

```
 {

test_hist, test_create, test_load, test_append, test_random, test_mod,
                test_write, test_write_prob,


                NULL, }
```

Referenced by count_tests(), and test().

Index

abs_double
    mcmc_internal.h, 82
accept
    mcmc, 8
ACCURACY_DEVIATION_FACTOR
    markov_chain.h, 47
ADAPT
    parallel_tempering.c, 87
adapt
    parallel_tempering.c, 87
additional_data
    mcmc, 8
analyse.c
    analyse_data_probability, 27
    analyse_marginal_distributions, 27
    calc_marginal_distribution, 27
    calc_mcmc_error, 27
    GNUPLOT_STYLE, 26
    HISTOGRAMS_MINMAX, 26
    NBINS, 26
analyse.h
    analyse_data_probability, 28
    analyse_marginal_distributions, 28
analyse_data_probability
    analyse.c, 27
    analyse.h, 28
    parallel_tempering.h, 90
analyse_marginal_distributions
    analyse.c, 27
    analyse.h, 28
    parallel_tempering.h, 90
append_to_hists
    histogram.c, 41
    histogram.h, 42
apply_formula
    simplesin.c, 22
    simplesin2.c, 24
    simplesin5.c, 25
apps/benchmark_main.c, 13
apps/bernoulli_example.c, 13
apps/eval_main.c, 15
apps/generic_main.c, 15
apps/library.c, 17
apps/normal.c, 19
apps/pulse.c, 19
apps/pulse_vrot.c, 21
apps/simplesin.c, 22
apps/simplesin2.c, 23
apps/simplesin5.c, 24
ASSERT
    tests.c, 107
assert
    mcmc.h, 56
ASSERTDUMP
    tests.c, 107
ASSERTEQUALD
    tests.c, 107
ASSERTEQUALI
    tests.c, 107
assess_acceptance_rate
    markov_chain.c, 44
    markov_chain.h, 48
ASSURE_DUMP_ENABLED
    mcmc_dump.c, 62
AT
    debug.h, 30

benchmark_main.c
    main, 13
bernoulli_example.c
    calc_model, 14
    calc_model_for, 14
    SIGMA, 14
beta
    parallel_tempering_mcmc, 11
BETA_0
    define_defaults.h, 33
BETA_0_STEPWIDTH
    parallel_tempering_beta.h, 94
BETA_ALIGNMENT
    parallel_tempering_beta.h, 94
BETWEEN

markov_chain_calibrate.c, 51
burn_in
markov_chain.c, 44
markov_chain.h, 49
BURN_IN_ITERATIONS
parallel_tempering.c, 87

calc_beta_0
parallel_tempering_beta.c, 92
parallel_tempering_beta.h, 94
calc_deviation
gsl_helper.c, 35
gsl_helper.h, 38
calc_hist
histogram.c, 41
histogram.h, 42
calc_marginal_distribution
analyse.c, 27
calc_mcmc_error
analyse.c, 27
calc_model
bernoulli_example.c, 14
library.c, 18
mcmc.h, 57
normal.c, 19
pulse.c, 20
pulse_vrot.c, 21
simplesin.c, 22
simplesin2.c, 24
simplesin5.c, 25
tests.c, 108
calc_model_for
bernoulli_example.c, 14
library.c, 18
mcmc.h, 57
normal.c, 19
pulse.c, 20
pulse_vrot.c, 21
simplesin.c, 23
simplesin2.c, 24
simplesin5.c, 25
tests.c, 108
calc_normalized
gsl_helper.c, 35
gsl_helper.h, 38
calc_prob
tests.c, 108
calc_same
gsl_helper.c, 35
gsl_helper.h, 38
calc_vector_squaresum
gsl_helper.c, 35
gsl_helper.h, 38
calc_vector_sum
gsl_helper.c, 35
gsl_helper.h, 38
calibrate_first
parallel_tempering.c, 87
parallel_tempering.h, 90
calibrate_rest
parallel_tempering.c, 87
parallel_tempering.h, 90
CALIBRATION_FILE
parallel_tempering.h, 89
parallel_tempering_config.h, 97
chebyshev_beta
parallel_tempering_beta.c, 92
chebyshev_stepwidth
parallel_tempering_beta.c, 92
chebyshev_temperature
parallel_tempering_beta.c, 92
check
generic_main.c, 16
checkfile
generic_main.c, 16
CIRCULAR_PARAMS
markov_chain.h, 47
tests.c, 108
clear_bit
markov_chain.c, 45
count_tests
run-tests.c, 105
tests.c, 108
countlines
mcmc_internal.h, 82
utils.c, 103
utils.h, 104
create_hist
histogram.c, 41
histogram.h, 42
ctrl_c_handler
parallel_tempering_run.c, 101

data
mcmc, 8
DATA_FILENAME
define_defaults.h, 33
DEBUG
debug.h, 30
debug

debug.h, 30
debug.c
dump_mcmc, 28
dump_vector, 28
dump_vectorln, 28
debug.h
AT, 30
DEBUG, 30
debug, 30
dump_d, 30
dump_i, 30
dump_i_s, 30
dump_l, 30
dump_m, 30
dump_mcmc, 32
dump_p, 30
dump_s, 30
dump_size, 31
dump_ui, 31
dump_ul, 31
dump_v, 31
dump_vector, 32
dump_vectorln, 32
IFDEBUG, 31
IFSEGV, 31
IFVERBOSE, 31
IFWAIT, 31
require, 31
SEGV, 31
STRINGIFY, 32
TOSTRING, 32
VERBOSE, 32
DEFAULT_ADJUST_STEP
markov_chain.h, 47
define_defaults.h
BETA_0, 33
DATA_FILENAME, 33
ITER_LIMIT, 33
MAX_AR_DEVIATION, 33
MUL, 33
N_BETA, 33
N_SWAP, 34
PARAMS_FILENAME, 34
TARGET_ACCEPTANCE_RATE, 34
do_step_for
markov_chain.c, 45
dump
parallel_tempering.c, 88
DUMP_ALL_CHAINS
parallel_tempering.h, 89
dump_d
debug.h, 30
DUMP_FORMAT
mcmc.h, 56
dump_i
debug.h, 30
dump_i_s
debug.h, 30
dump_l
debug.h, 30
dump_m
debug.h, 30
dump_mcmc
debug.c, 28
debug.h, 32
dump_p
debug.h, 30
dump_s
debug.h, 30
dump_size
debug.h, 31
dump_ui
debug.h, 31
dump_ul
debug.h, 31
dump_v
debug.h, 31
dump_vector
debug.c, 28
debug.h, 32
dump_vectorln
debug.c, 28
debug.h, 32
dumpflag
parallel_tempering_run.c, 101
parallel_tempering_run.h, 102
DUMPONFAIL
tests.c, 108
dup_vector
gsl_helper.c, 35
gsl_helper.h, 38

equidistant_beta
parallel_tempering_beta.c, 92
equidistant_stepwidth
parallel_tempering_beta.c, 92
equidistant_temperature
parallel_tempering_beta.c, 92
eval_main.c
main, 15

files
mcmc, 8
find_min_max
histogram.c, 41
histogram.h, 42
free_gsl_vector_array
mcmc_gettersetter.c, 66
FREEMSG
memory.h, 85

generic_main.c
check, 16
checkfile, 16
help_phase, 17
main, 17
MAX_ITERATIONS, 16
OUTPUT_PARAMD, 16
OUTPUT_PARAMI, 16
progname, 17
usage, 17
get_accept_rate
mcmc_gettersetter.c, 66
mcmc_gettersetter.h, 74
get_accept_rate_for
mcmc_gettersetter.c, 66
mcmc_gettersetter.h, 74
get_accept_rate_global
mcmc_gettersetter.c, 66
mcmc_gettersetter.h, 74
get_beta
parallel_tempering_beta.c, 92
parallel_tempering_beta.h, 94
get_bit
markov_chain.c, 45
get_chain_beta
parallel_tempering_beta.c, 92
parallel_tempering_beta.h, 95
get_column_count
utils.c, 103
utils.h, 104
get_data
mcmc_gettersetter.c, 66
get_duration
parallel_tempering_run.c, 101
parallel_tempering_run.h, 102
get_n_par
mcmc_gettersetter.c, 65
mcmc_gettersetter.h, 74
get_next_alog_urandom
mcmc_gettersetter.c, 66
mcmc_gettersetter.h, 75
get_next_random_jump
mcmc_gettersetter.c, 66
mcmc_gettersetter.h, 75
get_next_uniform_plusminus_random
mcmc_gettersetter.c, 67
mcmc_gettersetter.h, 75
get_next_uniform_random
mcmc_gettersetter.c, 67
mcmc_gettersetter.h, 75
get_params
mcmc_gettersetter.c, 67
mcmc_gettersetter.h, 75
get_params_accepts_for
mcmc_gettersetter.c, 67
mcmc_gettersetter.h, 76
get_params_accepts_global
mcmc_gettersetter.c, 67
mcmc_gettersetter.h, 76
get_params_accepts_sum
mcmc_gettersetter.c, 67
mcmc_gettersetter.h, 76
get_params_best
mcmc_gettersetter.c, 67
mcmc_gettersetter.h, 76
get_params_best_for
mcmc_gettersetter.c, 68
mcmc_gettersetter.h, 76
get_params_descr
mcmc_gettersetter.c, 68
mcmc_gettersetter.h, 76
get_params_for
mcmc_gettersetter.c, 68
mcmc_gettersetter.h, 76
get_params_max
mcmc_gettersetter.c, 68
mcmc_gettersetter.h, 76
get_params_max_for
mcmc_gettersetter.c, 68
mcmc_gettersetter.h, 77
get_params_min
mcmc_gettersetter.c, 68
mcmc_gettersetter.h, 77
get_params_min_for
mcmc_gettersetter.c, 68
mcmc_gettersetter.h, 77
get_params_rejects_for
mcmc_gettersetter.c, 68
mcmc_gettersetter.h, 77
get_params_rejects_global

mcmc_gettersetter.c, 69
mcmc_gettersetter.h, 77
get_params_rejects_sum
mcmc_gettersetter.c, 69
mcmc_gettersetter.h, 77
get_prior
mcmc_gettersetter.c, 69
mcmc_gettersetter.h, 77
get_prob
mcmc_gettersetter.c, 69
mcmc_gettersetter.h, 77
get_prob_best
mcmc_gettersetter.c, 69
mcmc_gettersetter.h, 78
get_random
mcmc_gettersetter.c, 69
mcmc_gettersetter.h, 78
get_steps
mcmc_gettersetter.c, 69
mcmc_gettersetter.h, 78
get_steps_for
mcmc_gettersetter.c, 70
mcmc_gettersetter.h, 78
get_steps_for_normalized
mcmc_gettersetter.c, 70
mcmc_gettersetter.h, 78
get_swapcount
parallel_tempering_beta.c, 93
parallel_tempering_beta.h, 95
get_ticks_per_second
parallel_tempering_run.c, 101
parallel_tempering_run.h, 102
get_vector_from_array
mcmc_gettersetter.c, 70
GNUPLOT_STYLE
analyse.c, 26
gsl_helper.c
calc_deviation, 35
calc_normalized, 35
calc_same, 35
calc_vector_squaresum, 35
calc_vector_sum, 35
dup_vector, 35
linreg_n, 36
max_column, 36
max_row, 36
max_vector, 36
min_column, 36
min_row, 36
min_vector, 36
sort, 37
xbar, 37
xbar_j, 37
gsl_helper.h
calc_deviation, 38
calc_normalized, 38
calc_same, 38
calc_vector_squaresum, 38
calc_vector_sum, 38
dup_vector, 38
linreg_n, 39
max_column, 39
max_row, 39
max_vector, 39
min_column, 39
min_row, 39
min_vector, 39
sort, 40

help_phase
generic_main.c, 17
histogram.c
append_to_hists, 41
calc_hist, 41
create_hist, 41
find_min_max, 41
update_min_max, 41
histogram.h
append_to_hists, 42
calc_hist, 42
create_hist, 42
find_min_max, 42
update_min_max, 42
HISTOGRAMS_MINMAX
analyse.c, 26
HMIN
pulse.c, 20
pulse_vrot.c, 21
hot_chains
parallel_tempering_beta.c, 93

IFDEBUG
debug.h, 31
IFDEBUGPARSER
mcmc_parser.c, 83
IFSEGV
debug.h, 31
IFVERBOSE
debug.h, 31
IFWAIT

debug.h, 31
inc_params_accepts
mcmc_gettersetter.c, 70
mcmc_gettersetter.h, 78
inc_params_accepts_for
mcmc_gettersetter.c, 70
mcmc_gettersetter.h, 78
inc_params_rejects
mcmc_gettersetter.c, 70
mcmc_gettersetter.h, 79
inc_params_rejects_for
mcmc_gettersetter.c, 70
mcmc_gettersetter.h, 79
inc_swapcount
parallel_tempering_beta.c, 93
parallel_tempering_beta.h, 95
init_seed
mcmc.c, 54
ITER_LIMIT
define_defaults.h, 33
ITER_READJUST
markov_chain.h, 47

library.c
calc_model, 18
calc_model_for, 18
p, 18
set_function, 18
linreg_n
gsl_helper.c, 36
gsl_helper.h, 39
LogLike
Problem, 11

main
benchmark_main.c, 13
eval_main.c, 15
generic_main.c, 17
run-tests.c, 105
markov_chain.c
assess_acceptance_rate, 44
burn_in, 44
clear_bit, 45
do_step_for, 45
get_bit, 45
markov_chain_step, 45
markov_chain_step_for, 45
MAXIMAL_STEPWIDTH, 44
MINIMAL_STEPWIDTH, 44
restart_from_best, 46
rmw_adapt_stepwidth, 46
set_bit, 46
markov_chain.h
ACCURACY_DEVIATION_FACTOR, 47
assess_acceptance_rate, 48
burn_in, 49
CIRCULAR_PARAMS, 47
DEFAULT_ADJUST_STEP, 47
ITER_READJUST, 47
markov_chain_calibrate, 49
markov_chain_step, 49
markov_chain_step_for, 50
NO_RESCALING_LIMIT, 47
restart_from_best, 50
rmw_adapt_stepwidth, 50
markov_chain_calibrate
markov_chain.h, 49
markov_chain_calibrate.c, 52
markov_chain_calibrate.c
BETWEEN, 51
markov_chain_calibrate, 52
markov_chain_calibrate_alt, 52
markov_chain_calibrate_linear_regression, 53
markov_chain_calibrate_multilinear_-regression, 53
markov_chain_calibrate_orig, 53
markov_chain_calibrate_quadratic, 53
MAX, 51
MAX_ACCURACY_IMPROVEMENT, 51
MIN, 51
SCALE_LIN_WORST, 52
SCALE_MIN, 52
markov_chain_calibrate_alt
markov_chain_calibrate.c, 52
markov_chain_calibrate_linear_regression
markov_chain_calibrate.c, 53
markov_chain_calibrate_multilinear_regression
markov_chain_calibrate.c, 53
markov_chain_calibrate_orig
markov_chain_calibrate.c, 53
markov_chain_calibrate_quadratic
markov_chain_calibrate.c, 53
markov_chain_step
markov_chain.c, 45
markov_chain.h, 49
markov_chain_step_for
markov_chain.c, 45

markov_chain.h, 50
MAX
markov_chain_calibrate.c, 51
MAX_ACCURACY_IMPROVEMENT
markov_chain_calibrate.c, 51
MAX_AR_DEVIATION
define_defaults.h, 33
max_column
gsl_helper.c, 36
gsl_helper.h, 39
MAX_ITERATIONS
generic_main.c, 16
MAX_LINE_LENGTH
mcmc_parser.c, 83
max_row
gsl_helper.c, 36
gsl_helper.h, 39
max_vector
gsl_helper.c, 36
gsl_helper.h, 39
MAXIMAL_STEPWIDTH
markov_chain.c, 44
mcmc, 7
accept, 8
additional_data, 8
data, 8
files, 8
n_iter, 8
n_par, 8
params, 8
params_accepts, 9
params_best, 9
params_descr, 9
params_max, 9
params_min, 9
params_rejects, 9
params_step, 10
prior, 10
prob, 10
prob_best, 10
random, 10
reject, 10
mcmc.c
init_seed, 54
mcmc_check, 54
mcmc_free, 54
mcmc_init, 55
r, 55
mcmc.h
assert, 56
calc_model, 57
calc_model_for, 57
DUMP_FORMAT, 56
mcmc_append_current_parameters, 57
mcmc_check, 58
mcmc_check_best, 58
mcmc_dump_close, 58
mcmc_dump_current, 58
mcmc_dump_flush, 58
mcmc_dump_probabilities, 58
mcmc_dump_y_dat, 59
mcmc_free, 59
mcmc_load, 59
mcmc_load_data, 60
mcmc_load_params, 60
mcmc_open_dump_files, 60
mcmc_reuse_data, 60
NOASSERT, 56
mcmc_append_current_parameters
mcmc.h, 57
mcmc_calculate.c, 61
mcmc_calculate.c
mcmc_append_current_parameters, 61
mcmc_check_best, 61
mcmc_check
mcmc.c, 54
mcmc.h, 58
mcmc_check_best
mcmc.h, 58
mcmc_calculate.c, 61
mcmc_dump.c
ASSURE_DUMP_ENABLED, 62
mcmc_dump_close, 63
mcmc_dump_current, 63
mcmc_dump_flush, 63
mcmc_dump_y_dat, 63
mcmc_open_dump_files, 63
mcmc_dump_close
mcmc.h, 58
mcmc_dump.c, 63
mcmc_dump_current
mcmc.h, 58
mcmc_dump.c, 63
mcmc_dump_flush
mcmc.h, 58
mcmc_dump.c, 63
mcmc_dump_probabilities
mcmc.h, 58
mcmc_dump_y_dat
mcmc.h, 59

mcmc_dump.c, 63
mcmc_free
mcmc.c, 54
mcmc.h, 59
mcmc_gettersetter.c
free_gsl_vector_array, 66
get_accept_rate, 66
get_accept_rate_for, 66
get_accept_rate_global, 66
get_data, 66
get_n_par, 65
get_next_alog_urandom, 66
get_next_random_jump, 66
get_next_uniform_plusminus_random, 67
get_next_uniform_random, 67
get_params, 67
get_params_accepts_for, 67
get_params_accepts_global, 67
get_params_accepts_sum, 67
get_params_best, 67
get_params_best_for, 68
get_params_descr, 68
get_params_for, 68
get_params_max, 68
get_params_max_for, 68
get_params_min, 68
get_params_min_for, 68
get_params_rejects_for, 68
get_params_rejects_global, 69
get_params_rejects_sum, 69
get_prior, 69
get_prob, 69
get_prob_best, 69
get_random, 69
get_steps, 69
get_steps_for, 70
get_steps_for_normalized, 70
get_vector_from_array, 70
inc_params_accepts, 70
inc_params_accepts_for, 70
inc_params_rejects, 70
inc_params_rejects_for, 70
N_PARAMETERS, 65
PROPOSAL, 66
reset_accept_rejects, 70
set_data, 71
set_minmax_for, 71
set_params, 71
set_params_accepts_for, 71
set_params_best, 71
set_params_descr_all, 71
set_params_descr_for, 71
set_params_for, 71
set_params_rejects_for, 72
set_prior, 72
set_prob, 72
set_prob_best, 72
set_random, 72
set_steps_all, 72
set_steps_for, 72
set_steps_for_normalized, 72
mcmc_gettersetter.h
get_accept_rate, 74
get_accept_rate_for, 74
get_accept_rate_global, 74
get_n_par, 74
get_next_alog_urandom, 75
get_next_random_jump, 75
get_next_uniform_plusminus_random, 75
get_next_uniform_random, 75
get_params, 75
get_params_accepts_for, 76
get_params_accepts_global, 76
get_params_accepts_sum, 76
get_params_best, 76
get_params_best_for, 76
get_params_descr, 76
get_params_for, 76
get_params_max, 76
get_params_max_for, 77
get_params_min, 77
get_params_min_for, 77
get_params_rejects_for, 77
get_params_rejects_global, 77
get_params_rejects_sum, 77
get_prior, 77
get_prob, 77
get_prob_best, 78
get_random, 78
get_steps, 78
get_steps_for, 78
get_steps_for_normalized, 78
inc_params_accepts, 78
inc_params_accepts_for, 78
inc_params_rejects, 79
inc_params_rejects_for, 79
reset_accept_rejects, 79
set_data, 79

set_minmax_for, 79
set_model, 79
set_n_par, 79
set_params, 79
set_params_accepts_for, 79
set_params_best, 80
set_params_descr_all, 80
set_params_descr_for, 80
set_params_for, 80
set_params_rejects_for, 80
set_prior, 80
set_prob, 80
set_prob_best, 80
set_random, 81
set_steps_all, 81
set_steps_for, 81
set_steps_for_normalized, 81
mcmc_init
mcmc.c, 55
mcmc_internal.h
abs_double, 82
countlines, 82
mod_double, 82
mcmc_load
mcmc.h, 59
mcmc_parser.c, 83
mcmc_load_data
mcmc.h, 60
mcmc_parser.c, 83
mcmc_load_params
mcmc.h, 60
mcmc_parser.c, 84
mcmc_open_dump_files
mcmc.h, 60
mcmc_dump.c, 63
mcmc_parser.c
IFDEBUGPARSER, 83
MAX_LINE_LENGTH, 83
mcmc_load, 83
mcmc_load_data, 83
mcmc_load_params, 84
mcmc_reuse_data, 84
my_strdup, 84
mcmc_reuse_data
mcmc.h, 60
mcmc_parser.c, 84
mem_calloc
memory.h, 85
mem_free
memory.h, 85
mem_malloc
memory.h, 85
mem_realloc
memory.h, 85
memory.h
FREEMSG, 85
mem_calloc, 85
mem_free, 85
mem_malloc, 85
mem_realloc, 85
WITHOUT_GARBAGE_COLLECTOR, 86
MIN
markov_chain_calibrate.c, 51
min_column
gsl_helper.c, 36
gsl_helper.h, 39
min_row
gsl_helper.c, 36
gsl_helper.h, 39
min_vector
gsl_helper.c, 36
gsl_helper.h, 39
MINIMAL_STEPWIDTH
markov_chain.c, 44
mod_double
mcmc_internal.h, 82
MUL
define_defaults.h, 33
my_strdup
mcmc_parser.c, 84

N_BETA
define_defaults.h, 33
n_iter
mcmc, 8
n_par
mcmc, 8
N_PARAMETERS
mcmc_gettersetter.c, 65
N_SWAP
define_defaults.h, 34
NBINS
analyse.c, 26
NO_RESCALING_LIMIT
markov_chain.h, 47
NOASSERT
mcmc.h, 56
normal.c
calc_model, 19

calc_model_for, 19

openfile
utils.c, 103
utils.h, 104
OUTPUT_PARAMD
generic_main.c, 16
OUTPUT_PARAMI
generic_main.c, 16

p
library.c, 18
parallel_tempering.c
ADAPT, 87
adapt, 87
BURN_IN_ITERATIONS, 87
calibrate_first, 87
calibrate_rest, 87
dump, 88
prepare_and_run_sampler, 88
register_signal_handlers, 88
report, 88
run_sampler, 88
RWM, 87
parallel_tempering.h
analyse_data_probability, 90
analyse_marginal_distributions, 90
calibrate_first, 90
calibrate_rest, 90
CALIBRATION_FILE, 89
DUMP_ALL_CHAINS, 89
prepare_and_run_sampler, 91
PRINT_PROB_INTERVAL, 89
SKIP_CALIBRATE_ALLCHAINS, 90
parallel_tempering_beta.c
calc_beta_0, 92
chebyshev_beta, 92
chebyshev_stepwidth, 92
chebyshev_temperature, 92
equidistant_beta, 92
equidistant_stepwidth, 92
equidistant_temperature, 92
get_beta, 92
get_chain_beta, 92
get_swapcount, 93
hot_chains, 93
inc_swapcount, 93
print_current_positions, 93
set_beta, 93
parallel_tempering_beta.h
BETA_0_STEPWIDTH, 94
BETA_ALIGNMENT, 94
calc_beta_0, 94
get_beta, 94
get_chain_beta, 95
get_swapcount, 95
inc_swapcount, 95
print_current_positions, 95
set_beta, 95
parallel_tempering_config.c
read_calibration_file, 96
setup_chains, 96
write_calibration_summary, 96
write_calibrations_file, 96
write_params_file, 96
parallel_tempering_config.h
CALIBRATION_FILE, 97
read_calibration_file, 97
setup_chains, 97
write_calibration_summary, 98
write_calibrations_file, 98
write_params_file, 98
parallel_tempering_decide_swap_nonrandom
parallel_tempering_interaction.c, 99
parallel_tempering_decide_swap_now
parallel_tempering_interaction.c, 99
parallel_tempering_decide_swap_random
parallel_tempering_interaction.c, 99
parallel_tempering_interaction.c
parallel_tempering_decide_swap_nonrandom, 99
parallel_tempering_decide_swap_now, 99
parallel_tempering_decide_swap_random, 99
tempering_interaction, 99
parallel_tempering_interaction.h
RANDOMSWAP, 100
tempering_interaction, 100
parallel_tempering_mcmc, 11
beta, 11
swapcount, 11
parallel_tempering_run.c
ctrl_c_handler, 101
dumpflag, 101
get_duration, 101
get_ticks_per_second, 101
register_signal_handlers, 101
run, 101
sigusr_handler, 101

parallel_tempering_run.h
    dumpflag, 102
    get_duration, 102
    get_ticks_per_second, 102
    register_signal_handlers, 102
    run, 102
params
    mcmc, 8
params_accepts
    mcmc, 9
params_best
    mcmc, 9
params_descr
    mcmc, 9
PARAMS_FILENAME
    define_defaults.h, 34
params_max
    mcmc, 9
params_min
    mcmc, 9
params_rejects
    mcmc, 9
params_step
    mcmc, 10
prepare_and_run_sampler
    parallel_tempering.c, 88
    parallel_tempering.h, 91
print_current_positions
    parallel_tempering_beta.c, 93
    parallel_tempering_beta.h, 95
PRINT_PROB_INTERVAL
    parallel_tempering.h, 89
Prior
    Problem, 11
prior
    mcmc, 10
prob
    mcmc, 10
prob_best
    mcmc, 10
Problem, 11
    LogLike, 11
    Prior, 11
progname
    generic_main.c, 17
PROPOSAL
    mcmc_gettersetter.c, 66
pulse.c
    calc_model, 20
    calc_model_for, 20
    HMIN, 20
pulse_vrot.c
    calc_model, 21
    calc_model_for, 21
    HMIN, 21

r
    mcmc.c, 55
random
    mcmc, 10
RANDOMSWAP
    parallel_tempering_interaction.h, 100
read_calibration_file
    parallel_tempering_config.c, 96
    parallel_tempering_config.h, 97
register_signal_handlers
    parallel_tempering.c, 88
    parallel_tempering_run.c, 101
    parallel_tempering_run.h, 102
reject
    mcmc, 10
report
    parallel_tempering.c, 88
require
    debug.h, 31
reset_accept_rejects
    mcmc_gettersetter.c, 70
    mcmc_gettersetter.h, 79
restart_from_best
    markov_chain.c, 46
    markov_chain.h, 50
rmw_adapt_stepwidth
    markov_chain.c, 46
    markov_chain.h, 50
run
    parallel_tempering_run.c, 101
    parallel_tempering_run.h, 102
run-tests.c
    count_tests, 105
    main, 105
    test, 105
    test_all, 105
    tests_registration, 105
    usage, 105
run_sampler
    parallel_tempering.c, 88
RWM
    parallel_tempering.c, 87

SCALE_LIN_WORST

markov_chain_calibrate.c, 52
SCALE_MIN
markov_chain_calibrate.c, 52
SEGV
debug.h, 31
set_beta
parallel_tempering_beta.c, 93
parallel_tempering_beta.h, 95
set_bit
markov_chain.c, 46
set_data
mcmc_gettersetter.c, 71
mcmc_gettersetter.h, 79
set_function
library.c, 18
set_minmax_for
mcmc_gettersetter.c, 71
mcmc_gettersetter.h, 79
set_model
mcmc_gettersetter.h, 79
set_n_par
mcmc_gettersetter.h, 79
set_params
mcmc_gettersetter.c, 71
mcmc_gettersetter.h, 79
set_params_accepts_for
mcmc_gettersetter.c, 71
mcmc_gettersetter.h, 79
set_params_best
mcmc_gettersetter.c, 71
mcmc_gettersetter.h, 80
set_params_descr_all
mcmc_gettersetter.c, 71
mcmc_gettersetter.h, 80
set_params_descr_for
mcmc_gettersetter.c, 71
mcmc_gettersetter.h, 80
set_params_for
mcmc_gettersetter.c, 71
mcmc_gettersetter.h, 80
set_params_rejects_for
mcmc_gettersetter.c, 72
mcmc_gettersetter.h, 80
set_prior
mcmc_gettersetter.c, 72
mcmc_gettersetter.h, 80
set_prob
mcmc_gettersetter.c, 72
mcmc_gettersetter.h, 80
set_prob_best
mcmc_gettersetter.c, 72
mcmc_gettersetter.h, 80
set_random
mcmc_gettersetter.c, 72
mcmc_gettersetter.h, 81
set_steps_all
mcmc_gettersetter.c, 72
mcmc_gettersetter.h, 81
set_steps_for
mcmc_gettersetter.c, 72
mcmc_gettersetter.h, 81
set_steps_for_normalized
mcmc_gettersetter.c, 72
mcmc_gettersetter.h, 81
setup_chains
parallel_tempering_config.c, 96
parallel_tempering_config.h, 97
SIGMA
bernoulli_example.c, 14
simplesin.c, 22
simplesin2.c, 23
simplesin5.c, 25
sigusr_handler
parallel_tempering_run.c, 101
simplesin.c
apply_formula, 22
calc_model, 22
calc_model_for, 23
SIGMA, 22
simplesin2.c
apply_formula, 24
calc_model, 24
calc_model_for, 24
SIGMA, 23
simplesin5.c
apply_formula, 25
calc_model, 25
calc_model_for, 25
SIGMA, 25
SKIP_CALIBRATE_ALLCHAINS
parallel_tempering.h, 90
sort
gsl_helper.c, 37
gsl_helper.h, 40
src/analyse.c, 26
src/analyse.h, 27
src/debug.c, 28
src/debug.h, 28
src/define_defaults.h, 33
src/gsl_helper.c, 34

src/gsl_helper.h, 37
src/histogram.c, 40
src/histogram.h, 42
src/markov_chain.c, 43
src/markov_chain.h, 46
src/markov_chain_calibrate.c, 50
src/mcmc.c, 54
src/mcmc.h, 55
src/mcmc_calculate.c, 61
src/mcmc_dump.c, 62
src/mcmc_gettersetter.c, 64
src/mcmc_gettersetter.h, 73
src/mcmc_internal.h, 81
src/mcmc_parser.c, 82
src/mcmc_struct.h, 84
src/memory.h, 85
src/parallel_tempering.c, 86
src/parallel_tempering.h, 89
src/parallel_tempering_beta.c, 91
src/parallel_tempering_beta.h, 93
src/parallel_tempering_config.c, 95
src/parallel_tempering_config.h, 97
src/parallel_tempering_interaction.c, 98
src/parallel_tempering_interaction.h, 99
src/parallel_tempering_run.c, 100
src/parallel_tempering_run.h, 102
src/utils.c, 103
src/utils.h, 103
STRINGIFY
    debug.h, 32
swapcount
    parallel_tempering_mcmc, 11

TARGET_ACCEPTANCE_RATE
    define_defaults.h, 34
tempering_interaction
    parallel_tempering_interaction.c, 99
    parallel_tempering_interaction.h, 100
test
    run-tests.c, 105
test_all
    run-tests.c, 105
test_append
    tests.c, 108
test_create
    tests.c, 109
test_hist
    tests.c, 109
test_load
    tests.c, 109
test_mod
    tests.c, 109
test_random
    tests.c, 109
test_tests
    tests.c, 109
test_write
    tests.c, 109
test_write_prob
    tests.c, 109
tests.c
    ASSERT, 107
    ASSERTDUMP, 107
    ASSERTEQUALD, 107
    ASSERTEQUALI, 107
    calc_model, 108
    calc_model_for, 108
    calc_prob, 108
    CIRCULAR_PARAMS, 108
    count_tests, 108
    DUMPONFAIL, 108
    test_append, 108
    test_create, 109
    test_hist, 109
    test_load, 109
    test_mod, 109
    test_random, 109
    test_tests, 109
    test_write, 109
    test_write_prob, 109
    tests_registration, 110
tests/run-tests.c, 104
tests/tests.c, 106
tests_registration
    run-tests.c, 105
    tests.c, 110
TOSTRING
    debug.h, 32

update_min_max
    histogram.c, 41
    histogram.h, 42
usage
    generic_main.c, 17
    run-tests.c, 105
utils.c
    countlines, 103
    get_column_count, 103
    openfile, 103
utils.h

countlines, 104
get_column_count, 104
openfile, 104

VERBOSE
debug.h, 32

WITHOUT_GARBAGE_COLLECTOR
memory.h, 86
write_calibration_summary
parallel_tempering_config.c, 96
parallel_tempering_config.h, 98
write_calibrations_file
parallel_tempering_config.c, 96
parallel_tempering_config.h, 98
write_params_file
parallel_tempering_config.c, 96
parallel_tempering_config.h, 98

xbar
gsl_helper.c, 37
xbar_j
gsl_helper.c, 37