

Paramfit

Generated by Doxygen 1.8.2

Mon Jun 24 2013 17:11:23

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	angle_data_struct Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	5
3.1.2.1	atom1	5
3.1.2.2	atom2	5
3.1.2.3	atom3	5
3.1.2.4	atom_type1	5
3.1.2.5	atom_type2	6
3.1.2.6	atom_type3	6
3.1.2.7	DO_FIT_KT	6
3.1.2.8	DO_FIT_THEQ	6
3.1.2.9	number	6
3.1.2.10	teq	6
3.1.2.11	tk	6
3.2	atom_struct Struct Reference	6
3.2.1	Detailed Description	6
3.2.2	Field Documentation	7
3.2.2.1	amass	7
3.2.2.2	chrg	7
3.2.2.3	iac	7
3.2.2.4	igraph	7
3.2.2.5	irotat	7
3.2.2.6	isymb1	7
3.2.2.7	itree	7
3.2.2.8	join	7

3.2.2.9	numex	7
3.2.2.10	res	7
3.3	bond_data_struct Struct Reference	7
3.3.1	Detailed Description	8
3.3.2	Field Documentation	8
3.3.2.1	atom1	8
3.3.2.2	atom2	8
3.3.2.3	atom_type1	8
3.3.2.4	atom_type2	8
3.3.2.5	DO_FIT_KR	8
3.3.2.6	DO_FIT_REQ	8
3.3.2.7	number	8
3.3.2.8	req	8
3.3.2.9	rk	8
3.4	bounds_struct Struct Reference	9
3.4.1	Detailed Description	9
3.4.2	Field Documentation	9
3.4.2.1	angle_thetas	9
3.4.2.2	bond_lengths	9
3.4.2.3	dihedral_thetas	9
3.4.2.4	mem_allocated	9
3.5	coord_set Struct Reference	9
3.5.1	Detailed Description	10
3.5.2	Field Documentation	10
3.5.2.1	energy_filename	10
3.5.2.2	filename	10
3.5.2.3	mem_allocated	10
3.5.2.4	natoms	10
3.5.2.5	num_coords	10
3.5.2.6	struc	10
3.6	coords_struct Struct Reference	10
3.6.1	Detailed Description	11
3.6.2	Field Documentation	11
3.6.2.1	energy	11
3.6.2.2	force	11
3.6.2.3	mem_allocated	11
3.6.2.4	x_coord	11
3.6.2.5	y_coord	11
3.6.2.6	z_coord	11
3.7	dihedral_data_struct Struct Reference	11

3.7.1	Detailed Description	12
3.7.2	Field Documentation	12
3.7.2.1	atom1	12
3.7.2.2	atom2	12
3.7.2.3	atom3	12
3.7.2.4	atom4	12
3.7.2.5	DO_FIT_KP	12
3.7.2.6	DO_FIT_NP	12
3.7.2.7	DO_FIT_PHASE	12
3.7.2.8	number	12
3.7.2.9	phase	12
3.7.2.10	pk	12
3.7.2.11	pn	12
3.8	dihedral_type_struct Struct Reference	12
3.8.1	Detailed Description	13
3.8.2	Field Documentation	13
3.8.2.1	atom_type1	13
3.8.2.2	atom_type2	13
3.8.2.3	atom_type3	13
3.8.2.4	atom_type4	13
3.8.2.5	improper	13
3.8.2.6	num_terms	13
3.8.2.7	term	13
3.9	force_struct Struct Reference	14
3.9.1	Field Documentation	14
3.9.1.1	x	14
3.9.1.2	y	14
3.9.1.3	z	14
3.10	global_options_struct Struct Reference	14
3.10.1	Detailed Description	15
3.10.2	Field Documentation	15
3.10.2.1	ALGORITHM	15
3.10.2.2	ANGLE_LIMIT	15
3.10.2.3	ANGLE_PARAMS	15
3.10.2.4	ANGLEEQ_dx	15
3.10.2.5	ANGLEFC_dx	16
3.10.2.6	BOND_LIMIT	16
3.10.2.7	BOND_PARAMS	16
3.10.2.8	BONDEQ_dx	16
3.10.2.9	BONDFC_dx	16

3.10.2.10 CHECK_BOUNDS	16
3.10.2.11 CONV_LIMIT	16
3.10.2.12 DIHEDRAL_PARAMS	16
3.10.2.13 DIHEDRAL_SPAN	16
3.10.2.14 DIHEDRALBH_dx	16
3.10.2.15 DIHEDRALG_dx	16
3.10.2.16 DIHEDRALN_dx	16
3.10.2.17 FIT_PHASE	16
3.10.2.18 FUNC_TO_FIT	17
3.10.2.19 GENERATIONS_TO_CONVERGE	17
3.10.2.20 job_control_filename	17
3.10.2.21 K	17
3.10.2.22 K_dx	17
3.10.2.23 K_FIT	17
3.10.2.24 MAX_GENERATIONS	17
3.10.2.25 mdcrd_list	17
3.10.2.26 mem_allocated	17
3.10.2.27 MUTATION_RATE	17
3.10.2.28 NDIHEDRALS	17
3.10.2.29 NDIMENSIONS	17
3.10.2.30 NOOPTIMIZATIONS	18
3.10.2.31 num_prmtops	18
3.10.2.32 PARAMETER_FILE_NAME	18
3.10.2.33 PARAMETERS_TO_FIT	18
3.10.2.34 PARENT_PERCENT	18
3.10.2.35 prmtop_list	18
3.10.2.36 QM_ENERGY_UNITS	18
3.10.2.37 QM_FORCE_UNITS	18
3.10.2.38 QM_SYSTEM_CHARGE	18
3.10.2.39 QM_SYSTEM_MULTIPLICITY	18
3.10.2.40 QMFILEFORMAT	18
3.10.2.41 QMFILEOUTEND	18
3.10.2.42 QMFILEOUTSTART	18
3.10.2.43 QMHEADER	18
3.10.2.44 RANDOM_SEED	19
3.10.2.45 RUNTYPE	19
3.10.2.46 SCATTERPLOTS	19
3.10.2.47 SCEE	19
3.10.2.48 SCNB	19
3.10.2.49 SEARCH_SPACE	19

3.10.2.50 TOTAL_STRUCTURES	19
3.10.2.51 VERBOSITY	19
3.10.2.52 WRITE_ENERGY	19
3.10.2.53 WRITE_FRCMOD	19
3.10.2.54 WRITE_PRMTOP	19
3.11 parm_struct Struct Reference	19
3.11.1 Detailed Description	21
3.11.2 Field Documentation	21
3.11.2.1 ag	21
3.11.2.2 AMBER_SYSTEM_CHARGE	21
3.11.2.3 angle_data	21
3.11.2.4 atom	21
3.11.2.5 bg	21
3.11.2.6 bond_data	21
3.11.2.7 cn1	21
3.11.2.8 cn2	22
3.11.2.9 dihedral_data	22
3.11.2.10 filename	22
3.11.2.11 fit_atom	22
3.11.2.12 hbcut	22
3.11.2.13 IFBOX	22
3.11.2.14 IFCAP	22
3.11.2.15 IFPERT	22
3.11.2.16 JHPARM	22
3.11.2.17 JPARM	22
3.11.2.18 MBONA	22
3.11.2.19 MBPER	22
3.11.2.20 MDPER	23
3.11.2.21 mem_allocated	23
3.11.2.22 MGPER	23
3.11.2.23 MPHIA	23
3.11.2.24 MPTRA	23
3.11.2.25 MTHETS	23
3.11.2.26 MUMANG	23
3.11.2.27 MUMBND	23
3.11.2.28 natex	23
3.11.2.29 NATYP	23
3.11.2.30 NBONA	23
3.11.2.31 NBONH	23
3.11.2.32 NBPER	24

3.11.2.33 ndimensions	24
3.11.2.34 NDPER	24
3.11.2.35 newparm	24
3.11.2.36 NEXT	24
3.11.2.37 NGPER	24
3.11.2.38 NHB	24
3.11.2.39 NMXRS	24
3.11.2.40 nno	24
3.11.2.41 NPHIA	24
3.11.2.42 NPHIH	24
3.11.2.43 NTHETA	24
3.11.2.44 NTHETH	25
3.11.2.45 NTOTAT	25
3.11.2.46 NTOTRS	25
3.11.2.47 NTYPES	25
3.11.2.48 NUMEXTRA	25
3.11.2.49 pangle	25
3.11.2.50 pangleH	25
3.11.2.51 pbond	25
3.11.2.52 pbondH	25
3.11.2.53 pdihedral	25
3.11.2.54 pdihedralH	25
3.11.2.55 phase	25
3.11.2.56 pk	26
3.11.2.57 pn	26
3.11.2.58 req	26
3.11.2.59 residue	26
3.11.2.60 rk	26
3.11.2.61 solty	26
3.11.2.62 teq	26
3.11.2.63 title	26
3.11.2.64 tk	26
3.11.2.65 unique_angles_found	26
3.11.2.66 unique_bonds_found	26
3.11.2.67 unique_dihedral_terms	26
3.11.2.68 unique_dihedrals_found	26
3.12 parmangle_struct Struct Reference	27
3.12.1 Detailed Description	27
3.12.2 Field Documentation	27
3.12.2.1 ict	27

3.12.2.2	it	27
3.12.2.3	jt	27
3.12.2.4	kt	27
3.13	parmbond_struct Struct Reference	27
3.13.1	Detailed Description	28
3.13.2	Field Documentation	28
3.13.2.1	ib	28
3.13.2.2	icb	28
3.13.2.3	jb	28
3.14	parmdihedral_struct Struct Reference	28
3.14.1	Detailed Description	28
3.14.2	Field Documentation	28
3.14.2.1	icp	28
3.14.2.2	ip	29
3.14.2.3	jp	29
3.14.2.4	kp	29
3.14.2.5	lp	29
3.15	residue Struct Reference	29
3.15.1	Detailed Description	29
3.15.2	Field Documentation	29
3.15.2.1	ipres	29
3.15.2.2	labres	29
4	File Documentation	31
4.1	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/bounds_check.c File Reference	31
4.1.1	Detailed Description	31
4.1.2	Function Documentation	31
4.1.2.1	calculate_structure_diversity	31
4.1.2.2	check_angles	32
4.1.2.3	check_bonds	32
4.1.2.4	check_dihedrals	32
4.1.2.5	check_range	33
4.1.2.6	clean_up_bounds	33
4.2	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/calc_r_squared.c File Reference	33
4.2.1	Function Documentation	34
4.2.1.1	calc_r_squared_multiprmtop	34
4.3	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/constants.h File Reference	34
4.3.1	Macro Definition Documentation	34
4.3.1.1	BOHR_TO_ANGSTROM	34
4.3.1.2	DEGREE_TO_RADIAN	34

4.3.1.3	HARTREE_TO_KCALMOL	34
4.3.1.4	KJMOL_TO_KCALMOL	34
4.3.1.5	PI	34
4.3.1.6	RADIAN_TO_DEGREE	34
4.4	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/create_input.c File Reference	34
4.4.1	Detailed Description	34
4.4.2	Function Documentation	35
4.4.2.1	create_input_single_prmtop	35
4.4.2.2	create_qm_input	35
4.5	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/defaults.c File Reference	35
4.5.1	Detailed Description	35
4.5.2	Function Documentation	36
4.5.2.1	set_default_options	36
4.6	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/dihedral_fitting.c File Reference	36
4.6.1	Detailed Description	36
4.6.2	Function Documentation	36
4.6.2.1	conduct_dihedral_least_squares	36
4.6.2.2	dihedral_least_squares	37
4.7	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/elements.c File Reference	37
4.7.1	Function Documentation	37
4.7.1.1	find_atomic_number_from_parm	37
4.7.1.2	print_atomic_number_as_symbol	37
4.8	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/error_messages.c File Reference	37
4.8.1	Detailed Description	38
4.8.2	Function Documentation	38
4.8.2.1	file_open_failure	38
4.8.2.2	malloc_failure_char	38
4.8.2.3	malloc_failure_double	38
4.8.2.4	malloc_failure_int	38
4.8.2.5	malloc_failure_short_int	39
4.8.2.6	process_retval	39
4.9	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/eval_amber_forces.c File Reference	39
4.9.1	Detailed Description	40
4.9.2	Function Documentation	40
4.9.2.1	eval_amber_forces_single_struct	40
4.9.2.2	eval_sum_amber_forces	40
4.9.2.3	eval_sum_amber_forces_multiprmtop	40
4.9.2.4	mark_relevant_atoms	41
4.9.2.5	print_forces	41
4.10	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/eval_amber_std.c File Reference	41

4.10.1 Detailed Description	42
4.10.2 Function Documentation	42
4.10.2.1 eval_amber_std_for_single_struct	42
4.10.2.2 eval_sum_squares_amber_std	42
4.10.2.3 eval_sum_squares_amber_std_multiprmtop	42
4.11 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/file_io.c File Reference	43
4.11.1 Detailed Description	43
4.11.2 Function Documentation	43
4.11.2.1 read_job_control_file	43
4.11.2.2 read_parameter_file	43
4.11.2.3 read_parameter_file_v2	44
4.12 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/fitting_control.c File Reference	44
4.12.1 Detailed Description	44
4.12.2 Function Documentation	44
4.12.2.1 do_fit	44
4.13 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/forces.h File Reference	45
4.14 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/function_def.h File Reference	45
4.14.1 Macro Definition Documentation	48
4.14.1.1 _GNU_SOURCE	48
4.14.1.2 ABORT	48
4.14.1.3 ALLOC_FAIL	48
4.14.1.4 ANGLES	48
4.14.1.5 BONDS	48
4.14.1.6 CMD_HELP_REQ	49
4.14.1.7 DATA_OVERFLOW	49
4.14.1.8 DEBUG	49
4.14.1.9 DIHEDRALS	49
4.14.1.10 EXCEEDEDMAXITERATIONS	49
4.14.1.11 FAILURE	49
4.14.1.12 FILE_OPEN_FAIL	49
4.14.1.13 FILE_READ_FAIL	49
4.14.1.14 HELP_REQ	49
4.14.1.15 HIST_REQ	49
4.14.1.16 INVALID_DATA	49
4.14.1.17 INVALID_FORMAT	49
4.14.1.18 INVALID_LINE	49
4.14.1.19 MINSTATIC	49
4.14.1.20 NOT_IMPLEMENTED	49
4.14.1.21 NSIMPLEX_INNER_PER_DIM	49
4.14.1.22 NSIMPLEX_OUTER_MAX	49

4.14.1.23	OFF	49
4.14.1.24	ON	49
4.14.1.25	RAND_RATIO	49
4.14.1.26	SUCCESS	49
4.14.1.27	TOO_MANY_OPT	49
4.14.1.28	UNKNOWN_ELEMENT	49
4.14.1.29	UNKNOWN_OPT	49
4.14.1.30	WARN	49
4.14.2	Enumeration Type Documentation	50
4.14.2.1	algorithm_t	50
4.14.2.2	bool_t	50
4.14.2.3	energy_t	50
4.14.2.4	force_t	50
4.14.2.5	function_t	50
4.14.2.6	parameter_mode_t	51
4.14.2.7	qm_format_t	51
4.14.2.8	readwrite_t	51
4.14.2.9	runtype_t	51
4.14.2.10	verbosity_t	51
4.14.3	Function Documentation	52
4.14.3.1	alloc_2D_double	52
4.14.3.2	alloc_coords	52
4.14.3.3	alloc_data_matrix	52
4.14.3.4	anglecomparator	52
4.14.3.5	bondcomparator	52
4.14.3.6	calc_angle_radians	52
4.14.3.7	calc_bond_length	53
4.14.3.8	calc_dihedral_radians	53
4.14.3.9	calc_fit_dimensions	53
4.14.3.10	calc_r_squared_multiprmtop	53
4.14.3.11	calculate_no_fit_params	54
4.14.3.12	calculate_structure_diversity	54
4.14.3.13	check_angles	54
4.14.3.14	check_bonds	55
4.14.3.15	check_dihedrals	55
4.14.3.16	check_for_valid_filename	55
4.14.3.17	check_range	56
4.14.3.18	clean_up_bounds	56
4.14.3.19	command_line_help	56
4.14.3.20	compare_energy	56

4.14.3.21	conduct_dihedral_least_squares	56
4.14.3.22	create_input_single_prmtop	57
4.14.3.23	create_qm_input	57
4.14.3.24	dihedral_least_squares	57
4.14.3.25	dihedral_types_equal	58
4.14.3.26	dihedralcomparator	58
4.14.3.27	do_fit	58
4.14.3.28	do_mutation	58
4.14.3.29	double_2D_array_free	59
4.14.3.30	eval_amber_forces_single_struct	59
4.14.3.31	eval_amber_std_for_single_struct	59
4.14.3.32	eval_sum_amber_forces	59
4.14.3.33	eval_sum_amber_forces_multiprmtop	60
4.14.3.34	eval_sum_squares_amber_std	60
4.14.3.35	eval_sum_squares_amber_std_multiprmtop	60
4.14.3.36	file_open_failure	61
4.14.3.37	find_atomic_number_from_parm	61
4.14.3.38	find_flag	61
4.14.3.39	free_coords	61
4.14.3.40	free_data_matrix	61
4.14.3.41	free_prmtop	62
4.14.3.42	genetic_wizard	62
4.14.3.43	get_float	62
4.14.3.44	get_option	62
4.14.3.45	global_unlock	62
4.14.3.46	handle_sigint	62
4.14.3.47	job_control_wizard	62
4.14.3.48	malloc_failure_char	63
4.14.3.49	malloc_failure_double	63
4.14.3.50	malloc_failure_int	63
4.14.3.51	malloc_failure_short_int	63
4.14.3.52	minimise_function_genetic	64
4.14.3.53	minimise_function_simplex	64
4.14.3.54	modify_params_scratch_data	64
4.14.3.55	name_copy	65
4.14.3.56	not_enough_dihedrals	65
4.14.3.57	ObfuscateAtom	65
4.14.3.58	print_atomic_number_as_symbol	65
4.14.3.59	print_backtrace	65
4.14.3.60	print_close_line_box	65

4.14.3.61	print_dihedral	66
4.14.3.62	print_forces	66
4.14.3.63	print_job_control_summary	66
4.14.3.64	print_multiprmtop_summary	66
4.14.3.65	print_open_line_box	67
4.14.3.66	print_parameter_summary	67
4.14.3.67	print_program_history	67
4.14.3.68	print_program_info	67
4.14.3.69	process_command_line	67
4.14.3.70	process_job_control_setting	68
4.14.3.71	process_prmtops	68
4.14.3.72	process_retval	68
4.14.3.73	process_single_prmtop	69
4.14.3.74	read_gaussian_energy	69
4.14.3.75	read_gaussian_forces	69
4.14.3.76	read_input_parameters	70
4.14.3.77	read_job_control_file	70
4.14.3.78	read_mdcrds	70
4.14.3.79	read_parameter_file	70
4.14.3.80	read_prmtops	71
4.14.3.81	read_qm	71
4.14.3.82	read_qm_directory	71
4.14.3.83	read_qm_energy_list	72
4.14.3.84	read_single_mdcrd	72
4.14.3.85	read_single_prmtop	72
4.14.3.86	s_getline	73
4.14.3.87	set_default_options	73
4.14.3.88	set_dihedral_fit	73
4.14.3.89	simplex_wizard	73
4.14.3.90	unObfuscateAtom	73
4.14.3.91	update_prmtop_data	74
4.14.3.92	verify_prmtops	74
4.14.3.93	write_energy	74
4.14.3.94	write_frcmod	75
4.14.3.95	write_input_adf	75
4.14.3.96	write_input_gamess	75
4.14.3.97	write_input_gaussian	76
4.14.3.98	write_input_parameters	76
4.14.3.99	write_prmtop	76
4.15	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/genetic_algorithm.c File Reference	77

4.15.1 Detailed Description	77
4.15.2 Function Documentation	77
4.15.2.1 alloc_data_matrix	77
4.15.2.2 do_mutation	78
4.15.2.3 free_data_matrix	78
4.15.2.4 minimise_function_genetic	78
4.16 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/help_functions.c File Reference	79
4.16.1 Function Documentation	79
4.16.1.1 command_line_help	79
4.17 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/mem_alloc.c File Reference	79
4.17.1 Function Documentation	79
4.17.1.1 alloc_2D_double	79
4.17.1.2 alloc_coords	79
4.17.1.3 double_2D_array_free	80
4.17.1.4 free_coords	80
4.17.1.5 free_prmtop	80
4.17.1.6 global_unlock	80
4.18 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/misc_utils.c File Reference	80
4.18.1 Detailed Description	81
4.18.2 Function Documentation	81
4.18.2.1 calc_angle_radians	81
4.18.2.2 calc_bond_length	81
4.18.2.3 calc_dihedral_radians	81
4.18.2.4 calc_fit_dimensions	82
4.18.2.5 calculate_no_fit_params	82
4.18.2.6 check_for_valid_filename	82
4.18.2.7 compare_energy	82
4.18.2.8 dihedral_types_equal	82
4.18.2.9 find_flag	83
4.18.2.10 handle_sigint	83
4.18.2.11 modify_params_scratch_data	83
4.18.2.12 name_copy	84
4.18.2.13 not_enough_dihedrals	84
4.18.2.14 ObfuscateAtom	84
4.18.2.15 print_backtrace	84
4.18.2.16 print_close_line_box	84
4.18.2.17 print_dihedral	85
4.18.2.18 print_open_line_box	85
4.18.2.19 s_getline	85
4.18.2.20 unObfuscateAtom	85

4.18.2.21	update_prmtop_data	86
4.19	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/options_summary.c File Reference	86
4.19.1	Detailed Description	86
4.19.2	Function Documentation	86
4.19.2.1	print_job_control_summary	86
4.20	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/param_summary.c File Reference	87
4.20.1	Detailed Description	87
4.20.2	Function Documentation	87
4.20.2.1	print_multiprmtop_summary	87
4.20.2.2	print_parameter_summary	87
4.21	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/parameter_optimiser.c File Reference	87
4.21.1	Function Documentation	88
4.21.1.1	main	88
4.22	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/print_program_info.c File Reference	88
4.22.1	Function Documentation	88
4.22.1.1	print_program_history	88
4.22.1.2	print_program_info	88
4.23	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/prmtop_params.h File Reference	88
4.23.1	Detailed Description	89
4.23.2	Macro Definition Documentation	89
4.23.2.1	BUFFER_SIZE	89
4.23.2.2	MAX_ANGLES_PER_TYPE	89
4.23.2.3	MAX_BONDS_PER_TYPE	89
4.23.2.4	MAX_DIHEDRALS_PER_TYPE	89
4.23.2.5	NAME_DEFAULT	89
4.23.2.6	NAME_SIZE	89
4.23.2.7	PRMTOP_TITLE_LENGTH	89
4.23.3	Typedef Documentation	89
4.23.3.1	Name	89
4.24	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/process_command_line.c File Reference	90
4.24.1	Detailed Description	90
4.24.2	Macro Definition Documentation	90
4.24.2.1	MAX_CMDLINE_OPTIONS	90
4.24.3	Function Documentation	90
4.24.3.1	process_command_line	90
4.25	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/process_job_control_setting.c File Reference	91
4.25.1	Detailed Description	91
4.25.2	Function Documentation	91
4.25.2.1	process_job_control_setting	91
4.26	/home/rbetz/git_tree/amber/AmberTools/src/paramfit/process_prmtop.c File Reference	92

4.26.1 Detailed Description	92
4.26.2 Function Documentation	92
4.26.2.1 anglecomparator	92
4.26.2.2 bondcomparator	92
4.26.2.3 dihedralcomparator	92
4.26.2.4 process_prmtops	93
4.26.2.5 process_single_prmtop	93
4.26.2.6 set_dihedral_fit	93
4.26.2.7 verify_prmtops	93
4.27 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/read_energy.c File Reference	94
4.27.1 Detailed Description	94
4.27.2 Function Documentation	94
4.27.2.1 read_gaussian_energy	94
4.27.2.2 read_gaussian_forces	95
4.27.2.3 read_qm	95
4.27.2.4 read_qm_directory	95
4.27.2.5 read_qm_energy_list	96
4.28 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/read_mdcrd.c File Reference	96
4.28.1 Detailed Description	96
4.28.2 Function Documentation	96
4.28.2.1 read_mdcrds	96
4.28.2.2 read_single_mdcrd	97
4.29 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/read_prmtop.c File Reference	97
4.29.1 Detailed Description	97
4.29.2 Function Documentation	97
4.29.2.1 read_prmtops	97
4.29.2.2 read_single_prmtop	98
4.30 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/simplex.c File Reference	98
4.30.1 Detailed Description	98
4.30.2 Function Documentation	98
4.30.2.1 minimise_function_simplex	98
4.31 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/wizard.c File Reference	99
4.31.1 Function Documentation	99
4.31.1.1 genetic_wizard	99
4.31.1.2 get_float	99
4.31.1.3 get_option	99
4.31.1.4 job_control_wizard	100
4.31.1.5 simplex_wizard	100
4.32 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/write_input.c File Reference	100
4.32.1 Detailed Description	100

4.32.2	Function Documentation	101
4.32.2.1	write_energy	101
4.32.2.2	write_frmmod	101
4.32.2.3	write_input_adf	101
4.32.2.4	write_input_gamess	101
4.32.2.5	write_input_gaussian	102
4.32.2.6	write_input_parameters	102
4.32.2.7	write_prmtp	102

Index**103**

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

angle_data_struct	5
atom_struct	6
bond_data_struct	7
bounds_struct	9
coord_set	9
coords_struct	10
dihedral_data_struct	11
dihedral_type_struct	12
force_struct	14
global_options_struct	14
parm_struct	19
parmangle_struct	27
parmbond_struct	27
parmdihedral_struct	28
residue	29

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

/home/rbetz/git_tree/amber/AmberTools/src/paramfit/bounds_check.c	31
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/calc_r_squared.c	33
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/constants.h	34
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/create_input.c	34
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/defaults.c	35
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/dihedral_fitting.c	36
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/elements.c	37
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/error_messages.c	37
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/eval_amber_forces.c	39
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/eval_amber_std.c	41
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/file_io.c	43
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/fitting_control.c	44
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/forces.h	45
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/function_def.h	45
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/genetic_algorithm.c	77
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/help_functions.c	79
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/mem_alloc.c	79
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/misc_utils.c	80
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/options_summary.c	86
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/param_summary.c	87
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/parameter_optimiser.c	87
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/print_program_info.c	88
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/prmtop_params.h	88
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/process_command_line.c	90
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/process_job_control_setting.c	91
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/process_prmtop.c	92
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/read_energy.c	94
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/read_mdcrd.c	96
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/read_prmtop.c	97
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/simplex.c	98
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/wizard.c	99
/home/rbetz/git_tree/amber/AmberTools/src/paramfit/write_input.c	100

Chapter 3

Data Structure Documentation

3.1 angle_data_struct Struct Reference

```
#include <prmtop_params.h>
```

Data Fields

- int [number](#)
- short int [DO_FIT_KT](#)
- short int [DO_FIT_THEQ](#)
- char [atom_type1](#) [[NAME_SIZE](#)]
- char [atom_type2](#) [[NAME_SIZE](#)]
- char [atom_type3](#) [[NAME_SIZE](#)]
- double [tk](#)
- double [teq](#)
- int [atom1](#) [[MAX_ANGLES_PER_TYPE](#)]
- int [atom2](#) [[MAX_ANGLES_PER_TYPE](#)]
- int [atom3](#) [[MAX_ANGLES_PER_TYPE](#)]

3.1.1 Detailed Description

Coordinates and parameters of all instances of one unique angle

3.1.2 Field Documentation

3.1.2.1 int angle_data_struct::atom1[[MAX_ANGLES_PER_TYPE](#)]

first The atoms involved in this angle type, - should really be a pointer here but this is quicker

3.1.2.2 int angle_data_struct::atom2[[MAX_ANGLES_PER_TYPE](#)]

3.1.2.3 int angle_data_struct::atom3[[MAX_ANGLES_PER_TYPE](#)]

3.1.2.4 char angle_data_struct::atom_type1[[NAME_SIZE](#)]

first The first atom type for this angle type

3.1.2.5 char angle_data_struct::atom_type2[NAME_SIZE]

first The second atom type for this angle type

3.1.2.6 char angle_data_struct::atom_type3[NAME_SIZE]

first The third atom type for this angle type

3.1.2.7 short int angle_data_struct::DO_FIT_KT

first Whether or not to vary this angle term in the fitting - default is yes

3.1.2.8 short int angle_data_struct::DO_FIT_THEQ

3.1.2.9 int angle_data_struct::number

first number of angles of this type

3.1.2.10 double angle_data_struct::teq

first Equilibrium angle for this angle type

3.1.2.11 double angle_data_struct::tk

first Force constant for this angle type

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

- [/home/rbetz/git_tree/amber/AmberTools/src/paramfit/prmtop_params.h](#)

3.2 atom_struct Struct Reference

```
#include <prmtop_params.h>
```

Data Fields

- double [chrg](#)
- double [amass](#)
- int [join](#)
- int [irotat](#)
- int [iac](#)
- int [numex](#)
- int [res](#)
- Name [igraph](#)
- Name [isymb1](#)
- Name [itree](#)

3.2.1 Detailed Description

Defines an atom with data from the prmtop

3.2.2 Field Documentation

3.2.2.1 double atom_struct::amass

the mass of the atom

3.2.2.2 double atom_struct::chrg

the charge on the atom

3.2.2.3 int atom_struct::iac

atom types involved in L-J

3.2.2.4 Name atom_struct::igraph

the true atom name

3.2.2.5 int atom_struct::irotat

last at to move if cur at moved

3.2.2.6 Name atom_struct::isymb1

the atom type

3.2.2.7 Name atom_struct::itree

the atom tree symbol

3.2.2.8 int atom_struct::join

the tree joining info

3.2.2.9 int atom_struct::numex

index into excluded atom list

3.2.2.10 int atom_struct::res

the residue number

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

- [/home/rbetz/git_tree/amber/AmberTools/src/paramfit/prmtop_params.h](#)

3.3 bond_data_struct Struct Reference

```
#include <prmtop_params.h>
```

Data Fields

- int [number](#)
- short int [DO_FIT_KR](#)
- short int [DO_FIT_REQ](#)
- char [atom_type1](#) [[NAME_SIZE](#)]
- char [atom_type2](#) [[NAME_SIZE](#)]
- double [rk](#)
- double [req](#)
- int [atom1](#) [[MAX_BONDS_PER_TYPE](#)]
- int [atom2](#) [[MAX_BONDS_PER_TYPE](#)]

3.3.1 Detailed Description

Coordinates and parameters of all instances of one unique bond

3.3.2 Field Documentation

3.3.2.1 int bond_data_struct::atom1[[MAX_BONDS_PER_TYPE](#)]

first The atoms involved in this bond type, - should really be a pointer here but this is quicker

3.3.2.2 int bond_data_struct::atom2[[MAX_BONDS_PER_TYPE](#)]

3.3.2.3 char bond_data_struct::atom_type1[[NAME_SIZE](#)]

first The first atom type for this bond type

3.3.2.4 char bond_data_struct::atom_type2[[NAME_SIZE](#)]

first The second atom type for this bond type

3.3.2.5 short int bond_data_struct::DO_FIT_KR

first Whether or not to vary this bond term in the fitting - default is yes

3.3.2.6 short int bond_data_struct::DO_FIT_REQ

3.3.2.7 int bond_data_struct::number

first number of bonds of this type

3.3.2.8 double bond_data_struct::req

first Equilibrium bond length for this bond type

3.3.2.9 double bond_data_struct::rk

first Force constant for this bond type

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

- [/home/rbetz/git_tree/amber/AmberTools/src/paramfit/prm_top_params.h](#)

3.4 bounds_struct Struct Reference

```
#include <function_def.h>
```

Data Fields

- double ** [bond_lengths](#)
- double ** [angle_thetas](#)
- double ** [dihedral_thetas](#)
- int [mem_allocated](#)

3.4.1 Detailed Description

Contains data used by the bounds checking functions. Only one of these is created.

3.4.2 Field Documentation

3.4.2.1 double** bounds_struct::angle_thetas

2D array of angles x all angle thetas found in the input conformations of the given angle

3.4.2.2 double** bounds_struct::bond_lengths

2D array of bonds x all bond lengths found in the input conformations of the given bond

3.4.2.3 double** bounds_struct::dihedral_thetas

2D array of dihedrals x all dihedral phases found in the input conformations of a given dihedral

3.4.2.4 int bounds_struct::mem_allocated

How much memory is being used

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

- [/home/rbetz/git_tree/amber/AmberTools/src/paramfit/function_def.h](#)

3.5 coord_set Struct Reference

```
#include <function_def.h>
```

Data Fields

- [coords_struct](#) * [struc](#)
- char * [filename](#)
- char * [energy_filename](#)
- int [num_coords](#)
- int [natoms](#)
- int [mem_allocated](#)

3.5.1 Detailed Description

Contains all structures for one input mdcrd. This is useful in multiprmtop fitting as having a 2D array of coord_structs introduces a lot more opportunities for bugs. This also helps keep track of the filename.

3.5.2 Field Documentation

3.5.2.1 `char* coord_set::energy_filename`

Name of the directory or file where qm output files/energies are

3.5.2.2 `char* coord_set::filename`

Name of the mdcrd this was read from

3.5.2.3 `int coord_set::mem_allocated`

Tracks amount of memory allocated in this structure

3.5.2.4 `int coord_set::natoms`

The number of atoms in each structure

3.5.2.5 `int coord_set::num_coords`

The number of structures in the array

3.5.2.6 `coords_struct* coord_set::struc`

Array containing one coordinate structure for each input conformation

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

- [/home/rbetz/git_tree/amber/AmberTools/src/paramfit/function_def.h](#)

3.6 coords_struct Struct Reference

```
#include <function_def.h>
```

Data Fields

- `int mem_allocated`
- `double * x_coord`
- `double * y_coord`
- `double * z_coord`
- `double energy`
- `force_struct * force`

3.6.1 Detailed Description

Contains data relating to one input conformation. One of these structures is created for each input conformation, and the array of these structures is collected in [coords_struct](#) *coords_data passed to most functions.

3.6.2 Field Documentation

3.6.2.1 double coords_struct::energy

QM energy for this coordinate set in Kcal/mol

3.6.2.2 force_struct* coords_struct::force

Contains forces on each atom if fitting forces

3.6.2.3 int coords_struct::mem_allocated

Updated for amount of memory allocated per structure

3.6.2.4 double* coords_struct::x_coord

Array of x-coordinates for each atom in the structure

3.6.2.5 double* coords_struct::y_coord

Array of y-coordinates for each atom in the structure

3.6.2.6 double* coords_struct::z_coord

Array of z-coordinates for each atom in the structure

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

- /home/rbetz/git_tree/amber/AmberTools/src/paramfit/[function_def.h](#)

3.7 dihedral_data_struct Struct Reference

```
#include <prmtop_params.h>
```

Data Fields

- int [number](#)
- short int [DO_FIT_KP](#)
- short int [DO_FIT_NP](#)
- short int [DO_FIT_PHASE](#)
- double [pk](#)
- double [pn](#)
- double [phase](#)
- int [atom1](#) [[MAX_DIHEDRALS_PER_TYPE](#)]
- int [atom2](#) [[MAX_DIHEDRALS_PER_TYPE](#)]
- int [atom3](#) [[MAX_DIHEDRALS_PER_TYPE](#)]
- int [atom4](#) [[MAX_DIHEDRALS_PER_TYPE](#)]

3.7.1 Detailed Description

Coordinates and parameters of all instances of one unique dihedral.

3.7.2 Field Documentation

3.7.2.1 `int dihedral_data_struct::atom1[MAX_DIHEDRALS_PER_TYPE]`

The atoms involved in this dihedral type, - should really be a pointer here but this is quicker

3.7.2.2 `int dihedral_data_struct::atom2[MAX_DIHEDRALS_PER_TYPE]`

3.7.2.3 `int dihedral_data_struct::atom3[MAX_DIHEDRALS_PER_TYPE]`

3.7.2.4 `int dihedral_data_struct::atom4[MAX_DIHEDRALS_PER_TYPE]`

3.7.2.5 `short int dihedral_data_struct::DO_FIT_KP`

Whether or not to vary this dihedral term in the fitting - default is yes for KP, no for NP and PHASE

3.7.2.6 `short int dihedral_data_struct::DO_FIT_NP`

3.7.2.7 `short int dihedral_data_struct::DO_FIT_PHASE`

3.7.2.8 `int dihedral_data_struct::number`

number of dihedrals of this type

3.7.2.9 `double dihedral_data_struct::phase`

phase of this dihedral type

3.7.2.10 `double dihedral_data_struct::pk`

Force constant for this dihedral type, note this is actually $V_n/2$

3.7.2.11 `double dihedral_data_struct::pn`

periodicity of this dihedral type

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

- `/home/rbetz/git_tree/amber/AmberTools/src/paramfit/prmtop_params.h`

3.8 `dihedral_type_struct` Struct Reference

```
#include <prmtop_params.h>
```

Data Fields

- char [atom_type1](#) [[NAME_SIZE](#)]
- char [atom_type2](#) [[NAME_SIZE](#)]
- char [atom_type3](#) [[NAME_SIZE](#)]
- char [atom_type4](#) [[NAME_SIZE](#)]
- int [num_terms](#)
- short int [improper](#)
- [dihedral_data_struct](#) * [term](#)

3.8.1 Detailed Description

All information about instances of one unique dihedral type Each term of this dihedral is a [dihedral_data_struct](#) array in term array This allows for easy addition of more terms to a dihedral and the identification of how many terms there are, etc.

3.8.2 Field Documentation

3.8.2.1 char dihedral_type_struct::atom_type1[NAME_SIZE]

The first atom type for this dihedral type

3.8.2.2 char dihedral_type_struct::atom_type2[NAME_SIZE]

The second atom type for this dihedral type

3.8.2.3 char dihedral_type_struct::atom_type3[NAME_SIZE]

The third atom type for this dihedral type

3.8.2.4 char dihedral_type_struct::atom_type4[NAME_SIZE]

The fourth atom type for this dihedral type

3.8.2.5 short int dihedral_type_struct::improper

whether or not this is an improper dihedral

3.8.2.6 int dihedral_type_struct::num_terms

The number of terms of this dihedral (# [dihedral_data_struct](#) arrays)

3.8.2.7 [dihedral_data_struct](#)* dihedral_type_struct::term

Array of dihedral data structures with data for each term

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

- [/home/rbetz/git_tree/amber/AmberTools/src/paramfit/prmtop_params.h](#)

3.9 force_struct Struct Reference

```
#include <forces.h>
```

Data Fields

- double [x](#)
- double [y](#)
- double [z](#)

3.9.1 Field Documentation

3.9.1.1 double [force_struct::x](#)

3.9.1.2 double [force_struct::y](#)

3.9.1.3 double [force_struct::z](#)

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

- [/home/rbetz/git_tree/amber/AmberTools/src/paramfit/forces.h](#)

3.10 global_options_struct Struct Reference

```
#include <function_def.h>
```

Data Fields

- int [mem_allocated](#)
- [verbosity_t](#) [VERBOSITY](#)
- char * [job_control_filename](#)
- char * [prmtop_list](#)
- char * [mdcrd_list](#)
- int [num_prmtops](#)
- char * [PARAMETER_FILE_NAME](#)
- char * [WRITE_ENERGY](#)
- char * [WRITE_FRCMOD](#)
- char * [WRITE_PRMTOP](#)
- int [RANDOM_SEED](#)
- [runtype_t](#) [RUNTYPE](#)
- [qm_format_t](#) [QMFILEFORMAT](#)
- [energy_t](#) [QM_ENERGY_UNITS](#)
- [force_t](#) [QM_FORCE_UNITS](#)
- [parameter_mode_t](#) [PARAMETERS_TO_FIT](#)
- [algorithm_t](#) [ALGORITHM](#)
- [function_t](#) [FUNC_TO_FIT](#)
- [bool_t](#) [K_FIT](#)
- double [K](#)
- int [BOND_PARAMS](#)
- int [ANGLE_PARAMS](#)
- int [DIHEDRAL_PARAMS](#)
- int [NDIHEDRALS](#)

- int [NOPTIMIZATIONS](#)
- int [MAX_GENERATIONS](#)
- int [GENERATIONS_TO_CONVERGE](#)
- double [SEARCH_SPACE](#)
- double [MUTATION_RATE](#)
- double [PARENT_PERCENT](#)
- double [CONV_LIMIT](#)
- double [BONDFC_dx](#)
- double [BONDEQ_dx](#)
- double [ANGLEFC_dx](#)
- double [ANGLEEQ_dx](#)
- double [DIHEDRALBH_dx](#)
- double [DIHEDRALN_dx](#)
- double [DIHEDRALG_dx](#)
- double [K_dx](#)
- [bool_t](#) [CHECK_BOUNDS](#)
- double [ANGLE_LIMIT](#)
- double [BOND_LIMIT](#)
- int [DIHEDRAL_SPAN](#)
- [bool_t](#) [SCATTERPLOTS](#)
- int [NDIMENSIONS](#)
- int [TOTAL_STRUCTURES](#)
- double [SCNB](#)
- double [SCEE](#)
- char * [QMFILEOUTSTART](#)
- char * [QMFILEOUTEND](#)
- char * [QMHEADER](#)
- int [QM_SYSTEM_CHARGE](#)
- int [QM_SYSTEM_MULTIPPLICITY](#)
- [bool_t](#) [FIT_PHASE](#)

3.10.1 Detailed Description

Contains all of the global options used by the program. Only one instance of this structure exists, and it is passed to most of the other functions and such

3.10.2 Field Documentation

3.10.2.1 [algorithm_t](#) [global_options_struct::ALGORITHM](#)

Fitting routine to be used

3.10.2.2 [double](#) [global_options_struct::ANGLE_LIMIT](#)

converged angle theta must be this close to values in input structures

3.10.2.3 [int](#) [global_options_struct::ANGLE_PARAMS](#)

3.10.2.4 [double](#) [global_options_struct::ANGLEEQ_dx](#)

simplex step sizes

3.10.2.5 `double global_options_struct::ANGLEFC_dx`

simplex step sizes

3.10.2.6 `double global_options_struct::BOND_LIMIT`

converged bond length must be this close to values in input structures

3.10.2.7 `int global_options_struct::BOND_PARAMS`

stores number of parameters of each type to be fit

3.10.2.8 `double global_options_struct::BONDEQ_dx`

simplex step sizes

3.10.2.9 `double global_options_struct::BONDFC_dx`

simplex step sizes

3.10.2.10 `bool_t global_options_struct::CHECK_BOUNDS`

whether or not to check bounds

3.10.2.11 `double global_options_struct::CONV_LIMIT`

convergence limit

3.10.2.12 `int global_options_struct::DIHEDRAL_PARAMS`

3.10.2.13 `int global_options_struct::DIHEDRAL_SPAN`

each dihedral must be spanned by this many input structures

3.10.2.14 `double global_options_struct::DIHEDRALBH_dx`

simplex step sizes. Dihedral BH in prmtop is actually $V_n/2$

3.10.2.15 `double global_options_struct::DIHEDRALG_dx`

simplex step sizes

3.10.2.16 `double global_options_struct::DIHEDRALN_dx`

simplex step sizes

3.10.2.17 `bool_t global_options_struct::FIT_PHASE`

Whether or not to fit dihedral phases in dihedral least squares function

3.10.2.18 function_t global_options_struct::FUNC_TO_FIT

The function to be used to fit to the energy surface

3.10.2.19 int global_options_struct::GENERATIONS_TO_CONVERGE

number generations in a row without a change to end algorithm

3.10.2.20 char* global_options_struct::job_control_filename

Contains the path and filename of the control file

3.10.2.21 double global_options_struct::K

intrinsic difference between quantum and classical

3.10.2.22 double global_options_struct::K_dx

simplex step sizes

3.10.2.23 bool_t global_options_struct::K_FIT

whether or not to fit the K parameter

3.10.2.24 int global_options_struct::MAX_GENERATIONS

maximum number of generations to run

3.10.2.25 char* global_options_struct::mdcrd_list

Path and filename of file containing list of mdcrd files

3.10.2.26 int global_options_struct::mem_allocated

Updated with number of bytes allocated for pointer inside this structure

3.10.2.27 double global_options_struct::MUTATION_RATE

percentage of values that may be mutated in a given generation

3.10.2.28 int global_options_struct::NDIHEDRALS

number of terms to give fitted dihedrals, at minimum

3.10.2.29 int global_options_struct::NDIMENSIONS

number of dimensions of fit, will be calculated

3.10.2.30 int global_options_struct::NOOPTIMIZATIONS

number of parameter sets per generation

3.10.2.31 int global_options_struct::num_prmtops

How many prmtops there are

3.10.2.32 char* global_options_struct::PARAMETER_FILE_NAME

Filename with parameters to be fit

3.10.2.33 parameter_mode_t global_options_struct::PARAMETERS_TO_FIT**3.10.2.34 double global_options_struct::PARENT_PERCENT**

percentage of values that are allowed to enter recombination

3.10.2.35 char* global_options_struct::prmtop_list

Path and filename of file containing list of prmtop files

3.10.2.36 energy_t global_options_struct::QM_ENERGY_UNITS

Unit of energy to expect from QM data file

3.10.2.37 force_t global_options_struct::QM_FORCE_UNITS

Unit of force from QM data file

3.10.2.38 int global_options_struct::QM_SYSTEM_CHARGE

integral charge of the system

3.10.2.39 int global_options_struct::QM_SYSTEM_MULTIPLICITY

integral multiplicity of the system

3.10.2.40 qm_format_t global_options_struct::QMFILEFORMAT**3.10.2.41 char* global_options_struct::QMFILEOUTEND****3.10.2.42 char* global_options_struct::QMFILEOUTSTART**

filename for QM input files- format is startNNNend where NNN is structure number

3.10.2.43 char* global_options_struct::QMHEADER

stores the location of a file to go at the beginning of qm input

3.10.2.44 int global_options_struct::RANDOM_SEED

for duplicating runs if necessary, for debugging usually

3.10.2.45 runtime_t global_options_struct::RUNTYPE**3.10.2.46 bool_t global_options_struct::SCATTERPLOTS**

whether or not to write scatter plots with input and output equilibrium parameters

3.10.2.47 double global_options_struct::SCEE**3.10.2.48 double global_options_struct::SCNB**

1-4 scaling factors for use with standard amber force field equation

3.10.2.49 double global_options_struct::SEARCH_SPACE

distance away from initial parameter set to search

3.10.2.50 int global_options_struct::TOTAL_STRUCTURES

number of input structures over all molecules

3.10.2.51 verbosity_t global_options_struct::VERBOSITY

How verbose to be - low, medium, or high

3.10.2.52 char* global_options_struct::WRITE_ENERGY

Filename, if any, to save final qm and md energies of structures to

3.10.2.53 char* global_options_struct::WRITE_FRCMOD

Filename, if any, to save frcmod to

3.10.2.54 char* global_options_struct::WRITE_PRMTOP

Filename, if any, to save a new prmtop to

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

- [/home/rbetz/git_tree/amber/AmberTools/src/paramfit/function_def.h](#)

3.11 parm_struct Struct Reference

```
#include <prmtop_params.h>
```

Data Fields

- int [mem_allocated](#)
- short int [newparm](#)
- char * [title](#)
- char * [filename](#)
- int [NTOTAT](#)
- int [NTYPES](#)
- int [NBONH](#)
- int [NBONA](#)
- int [NTHETH](#)
- int [NTHETA](#)
- int [NPHIH](#)
- int [NPHIA](#)
- int [JHPARM](#)
- int [JPARM](#)
- int [NEXT](#)
- int [NTOTRS](#)
- int [MBONA](#)
- int [MTHETS](#)
- int [MPHIA](#)
- int [MUMBND](#)
- int [MUMANG](#)
- int [MPTRA](#)
- int [NATYP](#)
- int [NHB](#)
- int [IFPERT](#)
- int [NBPER](#)
- int [NGPER](#)
- int [NDPER](#)
- int [MBPER](#)
- int [MGPER](#)
- int [MDPER](#)
- int [IFBOX](#)
- int [NMXRS](#)
- int [IFCAP](#)
- int [NUMEXTRA](#)
- [atom_struct](#) * [atom](#)
- double [AMBER_SYSTEM_CHARGE](#)
- int * [nno](#)
- [residue](#) * [residue](#)
- double * [rk](#)
- double * [req](#)
- double * [tk](#)
- double * [teq](#)
- double * [pk](#)
- double * [pn](#)
- double * [phase](#)
- double * [solty](#)
- double * [cn1](#)
- double * [cn2](#)
- [parmbond_struct](#) * [pbondH](#)
- [parmbond_struct](#) * [pbond](#)
- [parmangle_struct](#) * [pangleH](#)
- [parmangle_struct](#) * [pangle](#)

- [parmdihedral_struct](#) * [pdihedralH](#)
- [parmdihedral_struct](#) * [pdihedral](#)
- int * [natex](#)
- double * [ag](#)
- double * [bg](#)
- double * [hbcut](#)
- [bond_data_struct](#) * [bond_data](#)
- [angle_data_struct](#) * [angle_data](#)
- [dihedral_type_struct](#) * [dihedral_data](#)
- int [unique_bonds_found](#)
- int [unique_angles_found](#)
- int [unique_dihedrals_found](#)
- int [unique_dihedral_terms](#)
- int * [fit_atom](#)
- int [ndimensions](#)

3.11.1 Detailed Description

Stores all of the parmtop data

3.11.2 Field Documentation

3.11.2.1 double* parm_struct::ag

H-bond r^{*12} and r^{*10} ...

3.11.2.2 double parm_struct::AMBER.SYSTEM.CHARGE

Total charge of the system - calculated by summing the charges of the atoms

3.11.2.3 angle_data_struct* parm_struct::angle_data

Contains the angle data split into specific angle types

3.11.2.4 atom_struct* parm_struct::atom

Structure containing the info for each atom

3.11.2.5 double* parm_struct::bg

...

3.11.2.6 bond_data_struct* parm_struct::bond_data

Contains the bond data split into specific bond types

3.11.2.7 double* parm_struct::cn1

L-J r^{*12} and r^{*6} for all pos...

3.11.2.8 double* parm_struct::cn2

...atom type interactions

3.11.2.9 dihedral_type_struct* parm_struct::dihedral_data

Contains the dihedral data split into specific dihedral types

3.11.2.10 char* parm_struct::filename

The path to this file

3.11.2.11 int* parm_struct::fit_atom

Marks atoms as being relevant if forces are to be fit

3.11.2.12 double* parm_struct::hbcut

NO LONGER USED

3.11.2.13 int parm_struct::IFBOX

=1 if periodic box info to be read =0 otherwise

3.11.2.14 int parm_struct::IFCAP

=1 if CAP option was used in edit, =0 otherwise

3.11.2.15 int parm_struct::IFPERT

=1 if perturbation info is to be read =0 otherwise

3.11.2.16 int parm_struct::JHPARM

NOT USED

3.11.2.17 int parm_struct::JPARM

NOT USED

3.11.2.18 int parm_struct::MBONA

NBONA + number of constraint bonds

3.11.2.19 int parm_struct::MBPER

num of pert bonds across boundary to non-pert groups

3.11.2.20 int parm_struct::MDPER

num of pert dihedrals across bndry to non-pert groups

3.11.2.21 int parm_struct::mem_allocated

Updated with number of bytes allocated for pointer inside this structure

3.11.2.22 int parm_struct::MGPER

num of pert angles across boundary to non-pert groups

3.11.2.23 int parm_struct::MPHIA

NPHIA + number of constraint dihedral angles

3.11.2.24 int parm_struct::MPTRA

total number of unique dihedral types

3.11.2.25 int parm_struct::MTHETS

NTHETS (sic) + number of constraint angles

3.11.2.26 int parm_struct::MUMANG

total number of unique angle types

3.11.2.27 int parm_struct::MUMBND

total number of unique bond types

3.11.2.28 int* parm_struct::natex

excluded atom list

3.11.2.29 int parm_struct::NATYP

number of "atoms" defined in parameter file

3.11.2.30 int parm_struct::NBONA

number of bonds without hydrogen

3.11.2.31 int parm_struct::NBONH

number of bonds containing hydrogen

3.11.2.32 int parm_struct::NBPER

number of bonds to be perturbed

3.11.2.33 int parm_struct::ndimensions

The number of dimensions of fit in this structure

3.11.2.34 int parm_struct::NDPER

number of dihedrals to be perturbed

3.11.2.35 short int parm_struct::newparm

YES if the prmtop format is $\geq v7.0$ else NO

3.11.2.36 int parm_struct::NEXT

total number of excluded atoms

3.11.2.37 int parm_struct::NGPER

number of angles to be perturbed

3.11.2.38 int parm_struct::NHB

number of types of hydrogen bonded pair interactions

3.11.2.39 int parm_struct::NMXRS

number of atoms in the largest residue

3.11.2.40 int* parm_struct::nno

index for non-bond of each type

3.11.2.41 int parm_struct::NPHIA

number of dihedrals not containing hydrogen

3.11.2.42 int parm_struct::NPHIH

number of dihedrals containing hydrogen

3.11.2.43 int parm_struct::NTHETA

number of angles not containing hydrogen

3.11.2.44 int parm_struct::NTHETH

number of angles containing hydrogen

3.11.2.45 int parm_struct::NTOTAT

total number of atoms in the system

3.11.2.46 int parm_struct::NTOTRS

total number of residues

3.11.2.47 int parm_struct::NTYPES

number of AMBER atom types used, max is 60

3.11.2.48 int parm_struct::NUMEXTRA

number of extra points (aka lone pairs)

3.11.2.49 parmangle_struct* parm_struct::pangle

angles without hydrogen

3.11.2.50 parmangle_struct* parm_struct::pangleH

angles with hydrogen

3.11.2.51 parmbond_struct* parm_struct::pbond

bonds without hydrogen

3.11.2.52 parmbond_struct* parm_struct::pbondH

bonds with hydrogen

3.11.2.53 parmdihedral_struct* parm_struct::pdihedral

dihedrals without hydrogen

3.11.2.54 parmdihedral_struct* parm_struct::pdihedralH

dihedrals with hydrogen

3.11.2.55 double* parm_struct::phase

Dihedral phase constants

3.11.2.56 `double* parm_struct::pk`

dihedral force constants - note these are actually $vn/2$

3.11.2.57 `double* parm_struct::pn`

Dihedral periodicity constants

3.11.2.58 `double* parm_struct::req`

Bond equilibrium constants

3.11.2.59 `residue* parm_struct::residue`

Stores the residue info

3.11.2.60 `double* parm_struct::rk`

Bond force constants

3.11.2.61 `double* parm_struct::solty`

NOT USED BUT READ FROM PRMTOP ANYWAY

3.11.2.62 `double* parm_struct::teq`

Angle equilibrium constants

3.11.2.63 `char* parm_struct::title`

The title in the prmtop file

3.11.2.64 `double* parm_struct::tk`

Angle force constants

3.11.2.65 `int parm_struct::unique_angles_found`

3.11.2.66 `int parm_struct::unique_bonds_found`

3.11.2.67 `int parm_struct::unique_dihedral_terms`

Total number of unique dihedral terms

3.11.2.68 `int parm_struct::unique_dihedrals_found`

Total number of unique dihedral types. Each can have any number of terms

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

- /home/rbetz/git_tree/amber/AmberTools/src/paramfit/prmtop_params.h

3.12 parmangle_struct Struct Reference

```
#include <prmtop_params.h>
```

Data Fields

- int [it](#)
- int [jt](#)
- int [kt](#)
- int [ict](#)

3.12.1 Detailed Description

Unprocessed angle information as it appears in the prmtop Sorted for uniqueness and turned into a [angle_data_struct](#)

See Also

[process_prmtop](#)

3.12.2 Field Documentation

3.12.2.1 int parmangle_struct::ict

first pointer to parameters

3.12.2.2 int parmangle_struct::it

first first atom in angle

3.12.2.3 int parmangle_struct::jt

first second atom in angle

3.12.2.4 int parmangle_struct::kt

first third atom in angle

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

- [/home/rbetz/git_tree/amber/AmberTools/src/paramfit/prmtop_params.h](#)

3.13 parmbond_struct Struct Reference

```
#include <prmtop_params.h>
```

Data Fields

- int [ib](#)
- int [jb](#)
- int [icb](#)

3.13.1 Detailed Description

Unprocessed bond information as it appears in the prmtop Sorted for uniqueness and turned into a [bond_data_struct](#)

See Also

`process_prmtop`

3.13.2 Field Documentation

3.13.2.1 `int parmbond_struct::ib`

first atom in bond

3.13.2.2 `int parmbond_struct::icb`

first pointer to parameters

3.13.2.3 `int parmbond_struct::jb`

first second atom in bond

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

- `/home/rbetz/git_tree/amber/AmberTools/src/paramfit/prmtop_params.h`

3.14 `parmdihedral_struct` Struct Reference

```
#include <prmtop_params.h>
```

Data Fields

- `int ip`
- `int jp`
- `int kp`
- `int lp`
- `int icp`

3.14.1 Detailed Description

Unprocessed dihedral information as it appears in the prmtop Sorted for uniqueness and turned into a [dihedral_data_struct](#)

See Also

`process_prmtop`

3.14.2 Field Documentation

3.14.2.1 `int parmdihedral_struct::icp`

first pointer to parameters

3.14.2.2 int parmdihedral_struct::ip

first first atom in dihedral

3.14.2.3 int parmdihedral_struct::jp

first second atom in dihedral

3.14.2.4 int parmdihedral_struct::kp

first third atom in dihedral

3.14.2.5 int parmdihedral_struct::lp

first fourth atom in dihedral

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

- [/home/rbetz/git_tree/amber/AmberTools/src/paramfit/prmtop_params.h](#)

3.15 residue Struct Reference

```
#include <prmtop_params.h>
```

Data Fields

- [Name labres](#)
- [int ipres](#)

3.15.1 Detailed Description

Defines the residues in the prmtop.

3.15.2 Field Documentation

3.15.2.1 int residue::ipres

the pntr list or all residues

3.15.2.2 Name residue::labres

the residue name

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

- [/home/rbetz/git_tree/amber/AmberTools/src/paramfit/prmtop_params.h](#)

Chapter 4

File Documentation

4.1 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/bounds_check.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "constants.h"
#include "function_def.h"
```

Functions

- int [calculate_structure_diversity](#) ([global_options_struct](#) *global_options, [bounds_struct](#) *bounds_data, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- void [clean_up_bounds](#) ([bounds_struct](#) *bounds_data)
- int [check_dihedrals](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data, [bounds_struct](#) *bounds_data)
- int [check_angles](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data, [bounds_struct](#) *bounds_data)
- int [check_bonds](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data, [bounds_struct](#) *bounds_data)
- int [check_range](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data)

4.1.1 Detailed Description

Contains functions that ensure the minimization algorithms have not wandered off into areas where there are too little data in the input structures.

4.1.2 Function Documentation

4.1.2.1 int [calculate_structure_diversity](#) ([global_options_struct](#) * *global_options*, [bounds_struct](#) * *bounds_data*, [parm_struct](#) * *parm_data*, [coord_set](#) * *coords_data*)

Collects up information about bonds, angles, and dihedrals in the input structures in an easy to use data structure.

Parameters

in	<i>global_options</i>	The global options structure
out	<i>bounds_data</i>	The data structure with the data about input structure distributions
in	<i>parm_data</i>	Pointer to a single parameter data structure
in	<i>coords_data</i>	Pointer to a single input coordinates data structure

Returns

Integer indicating success or failure

4.1.2.2 `int check_angles (global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data, bounds_struct * bounds_data)`

Checks the angle equilibrium value is well defined in the input structures. This means that the value must be within ANGLE_LIMIT of an input structure angle value. This function is intended to be run with a converged set of parameters.

See Also

[calculate_structure_diversity](#)

Parameters

<code>in</code>	<code>global_options</code>	The global options structure
<code>in</code>	<code>parm_data</code>	Pointer to the parameter data structure, including the angle value to check
<code>in</code>	<code>coords_data</code>	Pointer to the coordinate set for these parameters
<code>in</code>	<code>bounds_data</code>	The table of bonds, angles, and dihedrals in the input structures

Returns

SUCCESS and a warning if out of bounds with check set to warn or ignore, else FAILURE

4.1.2.3 `int check_bonds (global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data, bounds_struct * bounds_data)`

Checks if the bond parameters are adequately represented in the input structures. The generated Keq should be within global_options->BOND_LIMIT of an input structure bond equilibrium distance. Returns SUCCESS or FAILURE.

See Also

[calculate_structure_diversity](#)

Parameters

<code>in</code>	<code>global_options</code>	The global options structure
<code>in</code>	<code>parm_data</code>	Pointer to the parameter structure with the bond parameters to check inside
<code>in</code>	<code>coords_data</code>	Pointer to coordinate structure that bounds data was gathered from
<code>in</code>	<code>bounds_data</code>	Table of bond, angle, and dihedral data in input structures

Returns

SUCCESS and prints a warning if bounds checking is set to ignore or warn, else FAILURE

4.1.2.4 `int check_dihedrals (global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data, bounds_struct * bounds_data)`

Checks the dihedral equilibrium phases are well represented in the input structures

See Also

[calculate_structure_diversity](#)

Parameters

in	<i>global_options</i>	The global options structure
in	<i>parm_data</i>	Pointer to single parameter struture containing the dihedral parameters
in	<i>coords_data</i>	Pointer to single coordinate set
in	<i>bounds_data</i>	The table of bond, angle, and dihedral values from the structures

Returns

SUCCESS and a warning if out of bounds for error or ignore checking options, else FAILURE

4.1.2.5 int check_range (global_options_struct * global_options, parm_struct * parm_data)

Checks that the bonds, angles, and dihedrals are in a valid range before conducting a function evaluation.

If they are not, it will change any invalid parameter to a random value within the valid range. This prevents the algorithms (especially the simplex algorithm) from crawling into corners of the valid solution space and getting stuck there because there is gradient that points into an invalid area.

Parameters

in	<i>global_options</i>	The global options structure
in, out	<i>parm_data</i>	The parameter structure

Returns

The number of parameters that were changed

4.1.2.6 void clean_up_bounds (bounds_struct * bounds_data)

Deletes the table of bond, angle, and dihedral structure data for clean up.

Parameters

in, out	<i>bounds_data</i>	The structure from which to delete the tables
---------	--------------------	---

4.2 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/calc_r_squared.c File Reference

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "constants.h"
#include "function_def.h"
```

Functions

- double [calc_r_squared_multiprmtop](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)

4.2.1 Function Documentation

4.2.1.1 `double calc_r_squared_multiprmtop (global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_data)`

4.3 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/constants.h File Reference

Macros

- `#define HARTREE_TO_KCALMOL 627.5098954`
- `#define KJMOL_TO_KCALMOL 1.0/4.184 /*Exact*/`
- `#define BOHR_TO_ANGSTROM 1.0/0.529`
- `#define PI 3.141592653589793238462643383279 /*OK it's only in fun:-)*/`
- `#define DEGREE_TO_RADIAN (2*PI)/360`
- `#define RADIAN_TO_DEGREE 360/(2*PI)`

4.3.1 Macro Definition Documentation

4.3.1.1 `#define BOHR_TO_ANGSTROM 1.0/0.529`

4.3.1.2 `#define DEGREE_TO_RADIAN (2*PI)/360`

4.3.1.3 `#define HARTREE_TO_KCALMOL 627.5098954`

4.3.1.4 `#define KJMOL_TO_KCALMOL 1.0/4.184 /*Exact*/`

4.3.1.5 `#define PI 3.141592653589793238462643383279 /*OK it's only in fun:-)*/`

4.3.1.6 `#define RADIAN_TO_DEGREE 360/(2*PI)`

4.4 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/create_input.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <sys/stat.h>
#include <stdlib.h>
#include "function_def.h"
```

Functions

- `int create_qm_input (global_options_struct *global_options, parm_struct *parm_datas, coord_set *coords_datas)`
- `int create_input_single_prmtop (global_options_struct *global_options, parm_struct *parm_data, coord_set *coords_data)`

4.4.1 Detailed Description

This file contains the routines for making the job files from a prmtop and mdcrd file that can then be used to obtain the energy data to be fitted against

4.4.2 Function Documentation

4.4.2.1 `int create_input_single_prmtop (global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data)`

Main input creating function for a single prmtop Sets up input files to write and calls the appropriate function to write it in the correct format.

Parameters

in	<i>global_options</i>	The global options structure
in	<i>parm_data</i>	Pointer to a single set of parameters for this molecule
in	<i>coords_data</i>	Pointer to a single coordinate set for this molecule

Returns

Integer indicating success or failure

4.4.2.2 `int create_qm_input (global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_datas)`

Main input creating function for multiple prmtops Creates input directories if non existent, creates the files to write in the dirctory, and calls the appropriate function to write the correct format. If there is only one molecule, falls through to old fashioned create_input for single prmtops.

Parameters

in	<i>global_options</i>	The global options structure
in	<i>parm_datas</i>	Pointer to the array of parm structures
in	<i>coords_datas</i>	Pointer to array of coordinate structures

Returns

Integer indicating success or failure

4.5 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/defaults.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "function_def.h"
#include "constants.h"
```

Functions

- `int set_default_options (global_options_struct *global_options)`

4.5.1 Detailed Description

Contains the default option for everything.

4.5.2 Function Documentation

4.5.2.1 `int set_default_options (global_options_struct * global_options)`

Initializes variables representing program options to their defaults.

Parameters

<code>in, out</code>	<code>global_options</code>	The global options structure where defaults will be set.
----------------------	-----------------------------	--

Returns

Integer indicating success or failure.

4.6 `/home/rbetz/git_tree/amber/AmberTools/src/paramfit/dihedral_fitting.c` File Reference

```
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "function_def.h"
#include "constants.h"
```

Functions

- `int dihedral_least_squares (global_options_struct *global_options, parm_struct *parm_datas, coord_set *coords_datas, bool_t fit_phase)`
- `int conduct_dihedral_least_squares (global_options_struct *global_options, parm_struct *parm_data, coord_set *coords_data, bool_t fit_phase)`

4.6.1 Detailed Description

Implements dihedral fitting as a linear minimization Uses the method described by Chad Hopkins and Adrian Roitberg to represent the dihedral fitting problem as linear minimization. Sets up the problem, constructs the linear problem, solves it with Cholesky decomposition and back-substitution, and returns new dihedral data

4.6.2 Function Documentation

4.6.2.1 `int conduct_dihedral_least_squares (global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data, bool_t fit_phase)`

Constructs and solves the linear system for the dihedral fitting algorithm.

Uses the transformations described by Hopkins and Roitberg to present dihedral fitting as a linear system and solves it using a Cholesky decomposition and back-substitution, then undoes the transformations to get back to normal dihedral space.

See Also

[dihedral_least_squares](#)

Parameters

<code>in</code>	<code>global_options</code>	The global options structure
<code>in, out</code>	<code>parm_data</code>	Array of input parameter structures, will be updated
<code>in</code>	<code>coords_data</code>	The array of input coordinate sets with associated QM energies
<code>in</code>	<code>fit_phase</code>	Whether to fit phases as well as dihedral force constants for Paramfit by Doxygen

Returns

Integer indicating success or failure

4.6.2.2 `int dihedral_least_squares (global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data, bool_t fit_phase)`

Wrapper function for all dihedral fitting functions.

Calls all the methods necessary to spit out new parameters so you don't have to remember which ones to use.

See Also

construct_initial_dihedral_params, [conduct_dihedral_least_squares](#)

Parameters

in	<i>global_options</i>	The global options structure
in, out	<i>parm_data</i>	Array containing the dihedral and parameter information at the beginning, and will be updated with the new parameters at the end.
in	<i>coords_data</i>	Pointer to single structure containing coordinates and QM energy of each input structure
in	<i>fit_phase</i>	Whether or not phases will be fit or just dihedral force constants

Returns

Integer indicating success or failure. *Parm_data* is updated with the new parameters if successful

4.7 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/elements.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "function_def.h"
```

Functions

- `int find_atomic_number_from_parm (parm_struct *parm_data, int atom)`
- `void print_atomic_number_as_symbol (FILE *fptr, int atomic_number)`

4.7.1 Function Documentation

4.7.1.1 `int find_atomic_number_from_parm (parm_struct * parm_data, int atom)`

4.7.1.2 `void print_atomic_number_as_symbol (FILE * fptr, int atomic_number)`

4.8 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/error_messages.c File Reference

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

Functions

- void [process_retval](#) (int err_code, [verbosity_t](#) VERBOSITY)
- void [malloc_failure_char](#) (char *routine, char *var_name, int chars_requested)
- void [malloc_failure_int](#) (char *routine, char *var_name, int ints_requested)
- void [malloc_failure_short_int](#) (char *routine, char *var_name, int short_ints_requested)
- void [malloc_failure_double](#) (char *routine, char *var_name, int doubles_requested)
- void [file_open_failure](#) (char *routine, char *var_name)

4.8.1 Detailed Description

Contains a number of functions for printing out error messages. These functions are usually called when a failure occurs, and they print out more information about the failure and then exit.

4.8.2 Function Documentation

4.8.2.1 void [file_open_failure](#) (char * *routine*, char * *var_name*)

Prints a standard file open failure message then exits. Includes the function name and the file that was attempted to be opened.

Parameters

in	<i>routine</i>	The function where the failure occurred
in	<i>var_name</i>	The filename that could not be opened

4.8.2.2 void [malloc_failure_char](#) (char * *routine*, char * *var_name*, int *chars_requested*)

Prints a standard malloc failure message for char data types then exits. Includes the routine name, variable name, and number of bytes requested for char data types

Parameters

in	<i>routine</i>	The function where the failure occurred
in	<i>var_name</i>	The variable that failed to be allocated
in	<i>chars_requested</i>	The number of characters requested to be allocated

4.8.2.3 void [malloc_failure_double](#) (char * *routine*, char * *var_name*, int *doubles_requested*)

Prints out a standard malloc failure message for double data types then exits. Includes the routine name, variable name, and number of bytes requested for double data types.

Parameters

in	<i>routine</i>	The function where the failure occurred
in	<i>var_name</i>	The variable that failed to be allocated
in	<i>doubles_requested</i>	The number of doubles that were to be allocated

4.8.2.4 void [malloc_failure_int](#) (char * *routine*, char * *var_name*, int *ints_requested*)

Prints out a standard malloc failure message for int data types then exits. Includes the routine name, variable name, and number of bytes requested for int data types.

Parameters

in	<i>routine</i>	The function where the failure occurred
in	<i>var_name</i>	The variable that failed to be allocated
in	<i>ints_requested</i>	The number of integers that were to be allocated

4.8.2.5 void malloc_failure_short_int (char * routine, char * var_name, int short_ints_requested)

Prints out a standard malloc failure message for short int data types then exits. Includes the routine name, variable name, and number of bytes requested for short int data types.

Parameters

in	<i>routine</i>	The function where the failure occurred
in	<i>var_name</i>	The variable that failed to be allocated
in	<i>short_ints_requested</i>	The number of short ints that were to be allocated

4.8.2.6 void process_retval (int err_code, verbosity_t VERBOSITY)

Processes the return value of a function and exits if an error

Parameters

in	<i>err_code</i>	The return value from the function
in	<i>VERBOSITY</i>	How verbose the program is

4.9 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/eval_amber_forces.c File Reference

```
#include "function_def.h"
#include "constants.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
```

Functions

- int [eval_amber_forces_single_struct](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coords_struct](#) *coords_data, [force_struct](#) *forces, int structure)
- double [eval_sum_amber_forces_multiprmtop](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- double [eval_sum_amber_forces](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [mark_relevant_atoms](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data)
- void [print_forces](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data, int atom)

4.9.1 Detailed Description

Functions for evaluating forces on each atom using the AMBER equation. This has been taken pretty much directly from the pmemd code.

4.9.2 Function Documentation

4.9.2.1 `int eval_amber_forces_single_struct (global_options_struct * global_options, parm_struct * parm_data, coords_struct * coords_data, force_struct * forces, int structure)`

Calculates forces on a single input structure. Uses the parameters in the parm struct and returns forces in the force struct for each atom.

Parameters

in	<i>global_options</i>	The global options structure
in	<i>parm_struct</i>	The parameter structure with force constants and equilibria to use
in	<i>coords_data</i>	Array of coordinate structures with atom positions
in, out	<i>forces</i>	Pre-allocated to array[NATOMS], will be populated with forces on each atom
in	<i>structure</i>	Index of the structure in coords_data to use for atom positions

Returns

Integer representing success or failure

4.9.2.2 `double eval_sum_amber_forces (global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data)`

Scores a set of parameters according to force comparison. Allocates all necessary force structures, runs the force calculation (in parallel with openmp) compares the result. The comparison is the average difference in force magnitude per structure.

Parameters

in	<i>global_options</i>	The global options structure
in	<i>parm_data</i>	The parameter set to evaluate
in	<i>coords_data</i>	Coordinate set containing atom coordinates for all input structures, and QM forces

Returns

Scalar representing average difference in force magnitude per structure.

4.9.2.3 `double eval_sum_amber_forces_multiprmtop (global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_datas)`

Scores a set of parameters cross multiple molecules according to force comparison. Essentially just sums the force evaluation value over all prmtops. This is safe to use if there is only one molecule so is recommended for all force evaluation calls.

Parameters

in	<i>global_options</i>	The global options structure
in	<i>parm_datas</i>	Array of parameter sets, one for each molecule
in	<i>coords_datas</i>	Array of coordinate data sets, one for each molecule

Returns

Double representing average difference in force magnitude over all molecules and structures

4.9.2.4 int mark_relevant_atoms (global_options_struct * global_options, parm_struct * parm_data)

Marks atoms that are involved in parameters to be fit. This is done when fitting forces so that only the forces on atoms involved in bonds, angles, or dihedrals to be fit are considered in the function evaluation. This marks those atoms so they may be easily accessed later.

Parameters

in	<i>global_options</i>	The global options structure
in, out	<i>parm_data</i>	The parameter data, including parameters to fit. Atoms will be marked in here.

Returns

Integer indicating success or failure

4.9.2.5 void print_forces (global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data, int atom)

Prints out a nice table with forces in amber and quantum for one atom over each structure. It will conduct the forces evaluation itself, including allocation of force structures.

Parameters

in	<i>global_options</i>	The global options structure
in	<i>parm_data</i>	The parameter set to use in the function
in	<i>coords_data</i>	Pointer to single set of atom coordinates in input structure
in	<i>atom</i>	Index of the atom to print out. 0 is usually a good choice.

4.10 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/eval_amber_std.c File Reference

```
#include <stdio.h>
#include <math.h>
#include "function_def.h"
#include "constants.h"
```

Functions

- double [eval_sum_squares_amber_std_multiprm_top](#) (global_options_struct *global_options, parm_struct *parm_data, coord_set *coords_data)
- double [eval_sum_squares_amber_std](#) (global_options_struct *global_options, parm_struct *parm_data, coord_set *coords_data)
- double [eval_amber_std_for_single_struct](#) (global_options_struct *global_options, parm_struct *parm_data, coords_struct *coords_data)

4.10.1 Detailed Description

Function evaluation code for energies with the AMBER equation. Evaluates the sum of the squares values using the standard AMBER force field, using the values in the parameter file for each structure in the coordinate data.

4.10.2 Function Documentation

4.10.2.1 `double eval_amber_std_for_single_struct (global_options_struct * global_options, parm_struct * parm_data, coords_struct * coords_data)`

Calculates the AMBER energy for a single structure.

Parameters

in	<i>global_options</i>	The global options structure
in	<i>parm_data</i>	Pointer to a single set of parameters to use in the calculation
in	<i>coords_data</i>	Pointer to a single coordinate structure to evaluate

Returns

The energy of the structure

4.10.2.2 `double eval_sum_squares_amber_std (global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data)`

Evaluates the sum squares difference between quantum and calculated energy for each structure. The energy calculation for each structure is done in parallel with openmp.

Parameters

in	<i>global_options</i>	The global options structure
in	<i>parm_data</i>	The parameter file containing parameter set to be scored
in	<i>coords_data</i>	Contains coordinates of atoms in all input structures, and qm energies (pointer to ONE coordinate set)

Returns

The sum of the squares of the difference in energy between AMBER function evaluation and quantum value.

4.10.2.3 `double eval_sum_squares_amber_std_multiprmtop (global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_datas)`

Conducts the energy evaluation over every prmtop.

Parameters

in	<i>global_options</i>	The global options structure
in	<i>parm_datas</i>	The array of parm structures
in	<i>coords_datas</i>	The array of coordinate sets

Returns

Sum of squares of energy difference over all structures.

4.11 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/file_io.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "function_def.h"
#include "constants.h"
```

Functions

- `int read_job_control_file (global_options_struct *global_options, coord_set *coords_data)`
- `int read_parameter_file_v2 (global_options_struct *global_options, parm_struct *parm_data)`
- `int read_parameter_file (global_options_struct *global_options, parm_struct *parm_data)`

4.11.1 Detailed Description

Responsible for all file input with the exception of processing the prmtop file.

See Also

[read_prmtop.c](#)

4.11.2 Function Documentation

4.11.2.1 `int read_job_control_file (global_options_struct * global_options, coord_set * coords_data)`

Reads the job control file and puts the options into the options structure.

Parameters

<code>in, out</code>	<code>global_options</code>	The global options structure, which contains the location of the job control file.
<code>in, out</code>	<code>coords_data</code>	Pointer to array of coords struct already allocated to size 1 that may have its number of

Returns

Integer indicating success or failure

4.11.2.2 `int read_parameter_file (global_options_struct * global_options, parm_struct * parm_data)`

Reads in a previously saved list of parameters to fit. The parameters are in the exact order as the prmtop, making this non-transferable between prmtops. This is now included for backwards-compatibility. It will check if you have a new format parameter file and call the v2 function to read it if found.

See Also

[read_parameter_file_v2](#)

Parameters

<code>in</code>	<code>global_options</code>	The global options structure, containing the file name
<code>in, out</code>	<code>parm_data</code>	The parameter data file, will be updated with parameters to fit.

Returns

Integer indicating success or failure.

4.11.2.3 `int read_parameter_file_v2 (global_options_struct * global_options, parm_struct * parm_dats)`

Reads in a previously saved list of parameters to fit. Parameters are listed by name, so can be used between and across prmtops.

Parameters

<code>in</code>	<code>global_options</code>	The global options structure, containing the file name
<code>in, out</code>	<code>parm_data</code>	The parameter data file, will be updated with parameters to fit.

Returns

Integer indicating success or failure.

4.12 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/fitting_control.c File Reference

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

Functions

- `int do_fit (global_options_struct *global_options, parm_struct *parm_dats, coord_set *coords_dats)`

4.12.1 Detailed Description

Contains the main routine that controls the fitting process, which calls the relevant algorithm after doing setup appropriate to the options the user has selected.

4.12.2 Function Documentation

4.12.2.1 `int do_fit (global_options_struct * global_options, parm_struct * parm_dats, coord_set * coords_dats)`

Conducts the fit. Prints a parameter summary, does bounds checking, calls appropriate algorithm, verifies and prints final parameters, and writes output file formats according to the options the user has chosen.

Parameters

<code>in</code>	<code>global_options</code>	The global options structure containing user defined options
<code>in, out</code>	<code>parm_data</code>	Array of initial parameters and what to fit for each molecule, will have final parameters inserted.
<code>in</code>	<code>coords_data</code>	Array of coordinate sets containing the atomic coordinates and QM data for input structures.

Returns

Integer indicating success or failure

4.13 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/forces.h File Reference

Data Structures

- struct [force_struct](#)

4.14 /home/rbetz/git_tree/amber/AmberTools/src/paramfit/function_def.h File Reference

```
#include "prmtop_params.h"
#include "forces.h"
#include <stdio.h>
```

Data Structures

- struct [global_options_struct](#)
- struct [coords_struct](#)
- struct [coord_set](#)
- struct [bounds_struct](#)

Macros

- #define [_GNU_SOURCE](#)
- #define [SUCCESS](#) -0
- #define [FAILURE](#) -1
- #define [NOT_IMPLEMENTED](#) -2
- #define [UNKNOWN_OPT](#) -101
- #define [TOO_MANY_OPT](#) -201
- #define [ALLOC_FAIL](#) -301
- #define [FILE_OPEN_FAIL](#) -401
- #define [FILE_READ_FAIL](#) -501
- #define [ABORT](#) -1000
- #define [CMD_HELP_REQ](#) -1001
- #define [HIST_REQ](#) -1002
- #define [HELP_REQ](#) -1003
- #define [INVALID_FORMAT](#) -2001
- #define [INVALID_DATA](#) -2002
- #define [INVALID_LINE](#) -2003
- #define [EXCEEDEDMAXITERATIONS](#) -3001
- #define [MINSTATIC](#) -3002
- #define [DATA_OVERFLOW](#) -3003
- #define [UNKNOWN_ELEMENT](#) -3004
- #define [OFF](#) 0
- #define [ON](#) 1
- #define [DEBUG](#) 2
- #define [WARN](#) 3
- #define [BONDS](#) 1
- #define [ANGLES](#) 2
- #define [DIHEDRALS](#) 3
- #define [NSIMPLEX_INNER_PER_DIM](#) 25 /*This is multiplied by the number of dimensions in order to work out how many inner loops to run*/
- #define [NSIMPLEX_OUTER_MAX](#) 100000
- #define [RAND_RATIO](#)

Enumerations

- enum `function_t` { `SUM_SQUARES_AMBER_STANDARD`, `AMBER_FORCES`, `DIHEDRAL_LEAST_SQUARES` }
- enum `algorithm_t` { `SIMPLEX`, `GENETIC`, `BOTH`, `NONE` }
- enum `qm_format_t` { `GAUSSIAN`, `ADF`, `GAMESS` }
- enum `bool_t` { `YES` = 1, `TRUE` = 1, `NO` = 0, `FALSE` = 0 }
- enum `readwrite_t` { `READ`, `WRITE` }
- enum `parameter_mode_t` { `DEFAULT`, `LOAD`, `SAVE`, `K_ONLY` }
- enum `runtype_t` { `CREATE_INPUT`, `FIT`, `SET_PARAMS` }
- enum `energy_t` { `HARTREE`, `KCALMOL`, `KJMOL` }
- enum `force_t` { `HARTREE_BOHR`, `KCALMOL_ANGSTROM` }
- enum `verbosity_t` { `LOW`, `MEDIUM`, `HIGH` }

Functions

- void `print_program_info` (void)
- void `print_program_history` (void)
- int `set_default_options` (`global_options_struct` *global_options)
- void `process_retval` (int err_code, `verbosity_t` VERBOSITY)
- void `malloc_failure_char` (char *routine, char *var_name, int chars_requested)
- void `malloc_failure_int` (char *routine, char *var_name, int ints_requested)
- void `malloc_failure_short_int` (char *routine, char *var_name, int short_ints_requested)
- void `malloc_failure_double` (char *routine, char *var_name, int doubles_requested)
- void `file_open_failure` (char *routine, char *var_name)
- double ** `alloc_2D_double` (int nrows, int ncolums)
- void `global_unlock` (`global_options_struct` *global_options, `parm_struct` **parm_data, `coord_set` **coords_data)
- void `print_close_line_box` (int no_spaces)
- void `print_open_line_box` (int *i)
- int `check_for_valid_filename` (const char *data_string, const int length)
- int `s_getline` (char *line, int max, FILE *fp)
- int `find_flag` (FILE *fptr, char *label)
- int `name_copy` (FILE *fptr, char *stringp)
- int `find_atomic_number_from_parm` (`parm_struct` *parm_data, int atom)
- void `print_atomic_number_as_symbol` (FILE *fptr, int atomic_number)
- double `calc_r_squared_multiprmtop` (`global_options_struct` *global_options, `parm_struct` *parm_data, `coord_set` *coords_data)
- int `unObfuscateAtom` (int at)
- int `ObfuscateAtom` (int at)
- void `print_parameter_summary` (`global_options_struct` *global_options, `parm_struct` *parm_data)
- double `calc_bond_length` (double bond1x, double bond1y, double bond1z, double bond2x, double bond2y, double bond2z)
- double `calc_angle_radians` (double atom1x, double atom1y, double atom1z, double atom2x, double atom2y, double atom2z, double atom3x, double atom3y, double atom3z)
- double `calc_dihedral_radians` (double atom1x, double atom1y, double atom1z, double atom2x, double atom2y, double atom2z, double atom3x, double atom3y, double atom3z, double atom4x, double atom4y, double atom4z)
- void `calc_fit_dimensions` (`global_options_struct` *global_options, `parm_struct` *parm_data)
- int `modify_params_scratch_data` (`global_options_struct` *global_options, `parm_struct` *parm_data, double *parameters, `readwrite_t` MODE)
- int `calculate_no_fit_params` (`parm_struct` *parm_data, short int MODE)
- void `double_2D_array_free` (double **array)
- int `write_input_parameters` (`global_options_struct` *global_options, `parm_struct` *parm_data)
- int `read_input_parameters` (`global_options_struct` *global_options, `parm_struct` *parm_data)

- int [read_parameter_file](#) (global_options_struct *global_options, [parm_struct](#) *parm_data)
- int [write_frcmod](#) (global_options_struct *global_options, [parm_struct](#) *parm_data)
- void [handle_sigint](#) (int param)
- void [print_backtrace](#) (int signal)
- int [dihedral_types_equal](#) (dihedral_type_struct *first, dihedral_type_struct *second)
- void [print_dihedral](#) (dihedral_type_struct *type, int term)
- int [compare_energy](#) (const void *a, const void *b)
- int [not_enough_dihedrals](#) ([parm_struct](#) *parm_data, int n)
- int [do_fit](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [minimise_function_simplex](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [check_range](#) (global_options_struct *global_options, [parm_struct](#) *parm_data)
- int [calculate_structure_diversity](#) (global_options_struct *global_options, [bounds_struct](#) *bounds_data, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [check_bonds](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data, [bounds_struct](#) *bounds_data)
- int [check_angles](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data, [bounds_struct](#) *bounds_data)
- int [check_dihedrals](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data, [bounds_struct](#) *bounds_data)
- void [clean_up_bounds](#) ([bounds_struct](#) *bounds_data)
- int [process_job_control_setting](#) (char *setting_line, int length, int *number_settings, global_options_struct *global_options, [coord_set](#) *coords_data)
- int [read_job_control_file](#) (global_options_struct *global_options, [coord_set](#) *coords_data)
- void [print_job_control_summary](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [process_command_line](#) (int argc, char **argv, global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- void [command_line_help](#) (char *cmd_line_options[])
- int [read_qm](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [read_qm_directory](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [read_qm_energy_list](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [read_gaussian_forces](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [read_gaussian_energy](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [write_input_adf](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *current_struct, int num, FILE *fptr)
- int [write_input_gamess](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *current_struct, int num, FILE *fptr)
- int [write_input_gaussian](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *current_struct, int num, FILE *fptr)
- int [create_qm_input](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [create_input_single_prmtop](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [write_energy](#) (global_options_struct *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data, int generation)
- void [set_dihedral_fit](#) (global_options_struct *global_options, dihedral_type_struct *s, int t, [bool_t](#) d_kp, [bool_t](#) d_np, [bool_t](#) d_phase)
- int [read_prmtops](#) (global_options_struct *global_options, [parm_struct](#) *parm_data)
- int [read_single_prmtop](#) (global_options_struct *global_options, [parm_struct](#) *parm_data)
- int [process_prmtops](#) (global_options_struct *global_options, [parm_struct](#) *parm_data)
- int [process_single_prmtop](#) (global_options_struct *global_options, [parm_struct](#) *parm_data)

- int [write_prmtop](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data)
- int [verify_prmtops](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_datas)
- int [bondcomparator](#) (const void *a, const void *b)
- int [anglecomparator](#) (const void *a, const void *b)
- int [dihedralcomparator](#) (const void *a, const void *b)
- void [print_multiprmtop_summary](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_datas)
- void [free_prmtop](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data)
- int [update_prmtop_data](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_datas, double *parameters)
- int [read_mdcrds](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_datas, [coord_set](#) **s_datas)
- int [read_single_mdcrd](#) ([coord_set](#) *coords_data)
- int [alloc_coords](#) ([global_options_struct](#) *global_options, [coord_set](#) *c)
- int [free_coords](#) ([global_options_struct](#) *global_options, [coord_set](#) *c)
- double [eval_amber_std_for_single_struct](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coords_struct](#) *coords_datas)
- double [eval_sum_squares_amber_std](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- double [eval_sum_squares_amber_std_multiprmtop](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_datas, [coord_set](#) *coords_datas)
- int [eval_amber_forces_single_struct](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coords_struct](#) *coords_data, [force_struct](#) *forces, int structure)
- double [eval_sum_amber_forces](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- void [print_forces](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data, int atom)
- double [eval_sum_amber_forces_multiprmtop](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_datas, [coord_set](#) *coords_datas)
- int [minimise_function_genetic](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data)
- int [do_mutation](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, double *row, int col, [bool_t](#) do_mutate)
- double ** [alloc_data_matrix](#) (int rows, int cols)
- void [free_data_matrix](#) (double **dm, int rows)
- int [job_control_wizard](#) ([global_options_struct](#) *global_options)
- int [get_option](#) (int min, int max)
- double [get_float](#) ()
- void [genetic_wizard](#) ([global_options_struct](#) *global_options, FILE *file)
- void [simplex_wizard](#) ([global_options_struct](#) *global_options, FILE *file)
- int [dihedral_least_squares](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data, [bool_t](#) fit_phase)
- int [conduct_dihedral_least_squares](#) ([global_options_struct](#) *global_options, [parm_struct](#) *parm_data, [coord_set](#) *coords_data, [bool_t](#) fit_phase)

4.14.1 Macro Definition Documentation

4.14.1.1 `#define GNU_SOURCE`

4.14.1.2 `#define ABORT -1000`

4.14.1.3 `#define ALLOC_FAIL -301`

4.14.1.4 `#define ANGLES 2`

4.14.1.5 `#define BONDS 1`

- 4.14.1.6 `#define CMD_HELP_REQ -1001`
 - 4.14.1.7 `#define DATA_OVERFLOW -3003`
 - 4.14.1.8 `#define DEBUG 2`
 - 4.14.1.9 `#define DIHEDRALS 3`
 - 4.14.1.10 `#define EXCEEDED_MAX_ITERATIONS -3001`
 - 4.14.1.11 `#define FAILURE -1`
 - 4.14.1.12 `#define FILE_OPEN_FAIL -401`
 - 4.14.1.13 `#define FILE_READ_FAIL -501`
 - 4.14.1.14 `#define HELP_REQ -1003`
 - 4.14.1.15 `#define HIST_REQ -1002`
 - 4.14.1.16 `#define INVALID_DATA -2002`
 - 4.14.1.17 `#define INVALID_FORMAT -2001`
 - 4.14.1.18 `#define INVALID_LINE -2003`
 - 4.14.1.19 `#define MIN_STATIC -3002`
 - 4.14.1.20 `#define NOT_IMPLEMENTED -2`
 - 4.14.1.21 `#define NSIMPLEX_INNER_PER_DIM 25 /*This is multiplied by the number of dimensions in order to work out how many inner loops to run*/`
 - 4.14.1.22 `#define NSIMPLEX_OUTER_MAX 100000`
 - 4.14.1.23 `#define OFF 0`
 - 4.14.1.24 `#define ON 1`
 - 4.14.1.25 `#define RAND_RATIO`
- Value:**
- ```
0.2 /*Ratio by which to multiply the random number received from rand() when
adjusting simplex vertices by gamma*/
```
- 4.14.1.26 `#define SUCCESS -0`
  - 4.14.1.27 `#define TOO_MANY_OPT -201`
  - 4.14.1.28 `#define UNKNOWN_ELEMENT -3004`
  - 4.14.1.29 `#define UNKNOWN_OPT -101`
  - 4.14.1.30 `#define WARN 3`

## 4.14.2 Enumeration Type Documentation

### 4.14.2.1 enum algorithm\_t

Available minimising algorithms

Enumerator:

***SIMPLEX***  
***GENETIC***  
***BOTH***  
***NONE***

### 4.14.2.2 enum bool\_t

Create a boolean type since this is in C

Enumerator:

***YES***  
***TRUE***  
***NO***  
***FALSE***

### 4.14.2.3 enum energy\_t

Units for energy available

Enumerator:

***HARTREE***  
***KCALMOL***  
***KJMOL***

### 4.14.2.4 enum force\_t

Units for force available

Enumerator:

***HARTREE\_BOHR***  
***KCALMOL\_ANGSTROM***

### 4.14.2.5 enum function\_t

Available minimising functions

Enumerator:

***SUM\_SQUARES\_AMBER\_STANDARD***  
***AMBER\_FORCES***  
***DIHEDRAL\_LEAST\_SQUARES***

## 4.14.2.6 enum parameter\_mode\_t

Settings for modes for parameter setting or getting

Enumerator:

**DEFAULT**  
**LOAD**  
**SAVE**  
**K\_ONLY**

## 4.14.2.7 enum qm\_format\_t

Input QM file formats enum

Enumerator:

**GAUSSIAN**  
**ADF**  
**GAMESS**

## 4.14.2.8 enum readwrite\_t

Read or write, used for file i/o

Enumerator:

**READ**  
**WRITE**

## 4.14.2.9 enum runtype\_t

Enum for paramfit's three run types

Enumerator:

**CREATE\_INPUT**  
**FIT**  
**SET\_PARAMS**

## 4.14.2.10 enum verbosity\_t

Level of verbosity available

Enumerator:

**LOW**  
**MEDIUM**  
**HIGH**

### 4.14.3 Function Documentation

4.14.3.1 `double** alloc_2D_double ( int nrows, int ncolumns )`

4.14.3.2 `int alloc_coords ( global_options_struct * global_options, coord_set * c )`

Allocates memory for one set of coordinates. Allocates a structure for each input conformation, and for each of those allocates space for the x,y,z coordinates of each atom.

#### Parameters

|                      |                             |                                                                                            |
|----------------------|-----------------------------|--------------------------------------------------------------------------------------------|
| <code>in</code>      | <code>global_options</code> | The global options structure                                                               |
| <code>in, out</code> | <code>c</code>              | Pointer to coordinate set to allocate for, with filename, natoms and nstructures specified |

#### Returns

Integer indicating success or failure

4.14.3.3 `double** alloc_data_matrix ( int rows, int cols )`

Allocates a more traditional, non-contiguous 2D array for the genetic algorithm. We don't use `alloc_2d_double` because that gets a contiguous block of memory and we specifically need non-contiguous since we will be swapping the rows around in sorting each generation, and free-ing the contiguous matrix later is a pain since it is now unclear which was the original first row.

#### See Also

[free\\_data\\_matrix](#)

#### Parameters

|                 |                   |                                          |
|-----------------|-------------------|------------------------------------------|
| <code>in</code> | <code>rows</code> | The number of rows to be in the array    |
| <code>in</code> | <code>cols</code> | The number of columns to be in the array |

#### Returns

Pointer to a 2D double array of the specified size

4.14.3.4 `int anglecomparator ( const void * a, const void * b )`

Compares names of two angles, for qsort

4.14.3.5 `int bondcomparator ( const void * a, const void * b )`

Compares names of two bonds, for qsort.

4.14.3.6 `double calc_angle_radians ( double atom1x, double atom1y, double atom1z, double atom2x, double atom2y, double atom2z, double atom3x, double atom3y, double atom3z )`

Calculates the angle between 3 3-dimensional points Angle between  $\vec{a}_1, \vec{a}_2, \vec{a}_3$  is defined as:  $\vec{v}_1 = \vec{a}_1 - \vec{a}_2$   $\vec{v}_2 = \vec{a}_3 - \vec{a}_2$

$$\theta = \cos^{-1} \frac{\vec{v}_1 \cdot \vec{v}_2}{|\vec{v}_1| |\vec{v}_2|}$$

## Returns

The angle between the three atoms

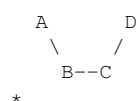
4.14.3.7 `double calc_bond_length ( double bond1x, double bond1y, double bond1z, double bond2x, double bond2y, double bond2z )`

Calculates the distance between 2 3-dimentional points.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

4.14.3.8 `double calc_dihedral_radians ( double atom1x, double atom1y, double atom1z, double atom2x, double atom2y, double atom2z, double atom3x, double atom3y, double atom3z, double atom4x, double atom4y, double atom4z )`

Calculates the dihedral between 4 points in 3 dimensions Given four atoms A,B,C & D:



The torsion angle along the torsion axis B-C is the angle between the planes ABC and BCD. The best way to calculate this is in terms of 3 vectors:

$\mathbf{a} = \mathbf{A} \rightarrow \mathbf{B}$   $\mathbf{b} = \mathbf{B} \rightarrow \mathbf{C}$   $\mathbf{c} = \mathbf{C} \rightarrow \mathbf{D}$

In terms of these vectors the dihedral angle is then:  $\theta = \cos^{-1} \frac{(\vec{a} \times \vec{b}) \cdot (\vec{b} \times \vec{c})}{|\vec{a} \times \vec{b}| |\vec{b} \times \vec{c}|}$

The sign of the dihedral is then found from the triple scalar product calculated by evaluating the determinant of the matrix:

$$\begin{vmatrix} a(x) & a(y) & a(z) \\ b(x) & b(y) & b(z) \\ c(x) & c(y) & c(z) \end{vmatrix} \rightarrow \begin{vmatrix} a(x) & a(y) & a(z) \\ b(x) & b(y) & b(z) \\ c(x) & c(y) & c(z) \end{vmatrix}$$

\*

which is:  $a[x] * (b[y]*c[z]-c[y]*b[z]) - a[y]*(b[x]*c[z]-c[x]*b[z]) + a[z]*(b[x]*c[y]-c[x]*b[y])$

## Returns

The angle between the input atoms, in radians

4.14.3.9 `void calc_fit_dimensions ( global_options_struct * global_options, parm_struct * parm_datas )`

Calculates the total number of parameters to be fit- the dimensionality of the problem. TODO - update with multiple prmtops as right now it is a simple loop

## Parameters

|                |                       |                                                                  |
|----------------|-----------------------|------------------------------------------------------------------|
| <i>in, out</i> | <i>global_options</i> | The global options structure, where NDIMENSIONS is to be updated |
| <i>in</i>      | <i>parm_datas</i>     | Array of parameter data structures                               |

4.14.3.10 `double calc_r_squared_multiptm ( global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_data )`

#### 4.14.3.11 int calculate\_no\_fit\_params ( parm\_struct \* parm\_data, short int MODE )

Calculates the number of parameters to be fit according to MODE Used to find number of bond, angle, and dihedral params separately.

##### Parameters

|    |                  |                                                                |
|----|------------------|----------------------------------------------------------------|
| in | <i>parm_data</i> | The parameter structure to examine                             |
| in | <i>MODE</i>      | BONDS, ANGLES, or DIHEDRALS indicating parameters to calculate |

##### Returns

The number of parameters of type MODE that are to be fit

#### 4.14.3.12 int calculate\_structure\_diversity ( global\_options\_struct \* global\_options, bounds\_struct \* bounds\_data, parm\_struct \* parm\_data, coord\_set \* coords\_data )

Collects up information about bonds, angles, and dihedrals in the input structures in an easy to use data structure.

##### Parameters

|     |                       |                                                                      |
|-----|-----------------------|----------------------------------------------------------------------|
| in  | <i>global_options</i> | The global options structure                                         |
| out | <i>bounds_data</i>    | The data structure with the data about input structure distributions |
| in  | <i>parm_data</i>      | Pointer to a single parameter data structure                         |
| in  | <i>coords_data</i>    | Pointer to a single input coordinates data structure                 |

##### Returns

Integer indicating success or failure

#### 4.14.3.13 int check\_angles ( global\_options\_struct \* global\_options, parm\_struct \* parm\_data, coord\_set \* coords\_data, bounds\_struct \* bounds\_data )

Checks the angle equilibrium value is well defined in the input structures. This means that the value must be within ANGLE\_LIMIT of an input structure angle value. This function is intended to be run with a converged set of parameters.

##### See Also

[calculate\\_structure\\_diversity](#)

##### Parameters

|    |                       |                                                                             |
|----|-----------------------|-----------------------------------------------------------------------------|
| in | <i>global_options</i> | The global options structure                                                |
| in | <i>parm_data</i>      | Pointer to the parameter data structure, including the angle value to check |
| in | <i>coords_data</i>    | Pointer to the coordinate set for these parameters                          |
| in | <i>bounds_data</i>    | The table of bonds, angles, and dihedrals in the input structures           |

##### Returns

SUCCESS and a warning if out of bounds with check set to warn or ignore, else FAILURE



4.14.3.14 `int check_bonds ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data, bounds_struct * bounds_data )`

Checks if the bond parameters are adequately represented in the input structures. The generated Keq should be within `global_options->BOND_LIMIT` of an input structure bond equilibrium distance. Returns SUCCESS or FAILURE.

See Also

[calculate\\_structure\\_diversity](#)

#### Parameters

|                 |                                    |                                                                             |
|-----------------|------------------------------------|-----------------------------------------------------------------------------|
| <code>in</code> | <code><i>global_options</i></code> | The global options structure                                                |
| <code>in</code> | <code><i>parm_data</i></code>      | Pointer to the parameter structure with the bond parameters to check inside |
| <code>in</code> | <code><i>coords_data</i></code>    | Pointer to coordinate structure that bounds data was gathered from          |
| <code>in</code> | <code><i>bounds_data</i></code>    | Table of bond, angle, and dihedral data in input structures                 |

#### Returns

SUCCESS and prints a warning if bounds checking is set to ignore or warn, else FAILURE

4.14.3.15 `int check_dihedrals ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data, bounds_struct * bounds_data )`

Checks the dihedral equilibrium phases are well represented in the input structures

See Also

[calculate\\_structure\\_diversity](#)

#### Parameters

|                 |                                    |                                                                         |
|-----------------|------------------------------------|-------------------------------------------------------------------------|
| <code>in</code> | <code><i>global_options</i></code> | The global options structure                                            |
| <code>in</code> | <code><i>parm_data</i></code>      | Pointer to single parameter struture containing the dihedral parameters |
| <code>in</code> | <code><i>coords_data</i></code>    | Pointer to single coordinate set                                        |
| <code>in</code> | <code><i>bounds_data</i></code>    | The table of bond, angle, and dihedral values from the structures       |

#### Returns

SUCCESS and a warning if out of bounds for error or ignore checking options, else FAILURE

4.14.3.16 `int check_for_valid_filename ( const char * data_string, const int length )`

Checks for invalid characters in a filename string.

#### Parameters

|                 |                                 |                          |
|-----------------|---------------------------------|--------------------------|
| <code>in</code> | <code><i>data_string</i></code> | The string to check      |
| <code>in</code> | <code><i>length</i></code>      | The length of the string |

#### Returns

Integer representing SUCCESS or INVALID\_FORMAT

#### 4.14.3.17 int check\_range ( global\_options\_struct \* global\_options, parm\_struct \* parm\_data )

Checks that the bonds, angles, and dihedrals are in a valid range before conducting a function evaluation.

If they are not, it will change any invalid parameter to a random value within the valid range. This prevents the algorithms (especially the simplex algorithm) from crawling into corners of the valid solution space and getting stuck there because there is gradient that points into an invalid area.

##### Parameters

|         |                       |                              |
|---------|-----------------------|------------------------------|
| in      | <i>global_options</i> | The global options structure |
| in, out | <i>parm_data</i>      | The parameter structure      |

##### Returns

The number of parameters that were changed

#### 4.14.3.18 void clean\_up\_bounds ( bounds\_struct \* bounds\_data )

Deletes the table of bond, angle, and dihedral structure data for clean up.

##### Parameters

|         |                    |                                               |
|---------|--------------------|-----------------------------------------------|
| in, out | <i>bounds_data</i> | The structure from which to delete the tables |
|---------|--------------------|-----------------------------------------------|

#### 4.14.3.19 void command\_line\_help ( char \* cmd\_line\_options[] )

#### 4.14.3.20 int compare\_energy ( const void \* a, const void \* b )

Compare function for qsort in the coordinate structures. Used to compare two coords structures by energy. Please don't call this directly, just pass a function pointer to qsort.

#### 4.14.3.21 int conduct\_dihedral\_least\_squares ( global\_options\_struct \* global\_options, parm\_struct \* parm\_data, coord\_set \* coords\_data, bool\_t fit\_phase )

Constructs and solves the linear system for the dihedral fitting algorithm.

Uses the transformations described by Hopkins and Roitberg to present dihedral fitting as a linear system and solves it using a Cholesky decomposition and back-substitution, then undoes the transformations to get back to normal dihedral space.

##### See Also

[dihedral\\_least\\_squares](#)

##### Parameters

|         |                       |                                                                |
|---------|-----------------------|----------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                   |
| in, out | <i>parm_data</i>      | Array of input parameter structures, will be updated           |
| in      | <i>coords_data</i>    | The array of input coordinate sets with associated QM energies |
| in      | <i>fit_phase</i>      | Whether to fit phases as well or just dihedral force constants |

##### Returns

Integer indicating success or failure

4.14.3.22 `int create_input_single_prmtop ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data )`

Main input creating function for a single prmtop Sets up input files to write and calls the appropriate function to write it in the correct format.

#### Parameters

|                 |                                    |                                                         |
|-----------------|------------------------------------|---------------------------------------------------------|
| <code>in</code> | <code><i>global_options</i></code> | The global options structure                            |
| <code>in</code> | <code><i>parm_data</i></code>      | Pointer to a single set of parameters for this molecule |
| <code>in</code> | <code><i>coords_data</i></code>    | Pointer to a single coordinate set for this molecule    |

#### Returns

Integer indicating success or failure

4.14.3.23 `int create_qm_input ( global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_datas )`

Main input creating function for multiple prmtops Creates input directories if non existent, creates the files to write in the directory, and calls the appropriate function to write the correct format. If there is only one molecule, falls through to old fashioned create\_input for single prmtops.

#### Parameters

|                 |                                    |                                           |
|-----------------|------------------------------------|-------------------------------------------|
| <code>in</code> | <code><i>global_options</i></code> | The global options structure              |
| <code>in</code> | <code><i>parm_datas</i></code>     | Pointer to the array of parm structures   |
| <code>in</code> | <code><i>coords_datas</i></code>   | Pointer to array of coordinate structures |

#### Returns

Integer indicating success or failure

4.14.3.24 `int dihedral_least_squares ( global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_datas, bool_t fit_phase )`

Wrapper function for all dihedral fitting functions.

Calls all the methods necessary to spit out new parameters so you don't have to remember which ones to use.

#### See Also

construct\_initial\_dihedral\_params, [conduct\\_dihedral\\_least\\_squares](#)

#### Parameters

|                      |                                    |                                                                                                                                   |
|----------------------|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <code>in</code>      | <code><i>global_options</i></code> | The global options structure                                                                                                      |
| <code>in, out</code> | <code><i>parm_data</i></code>      | Array containing the dihedral and parameter information at the beginning, and will be updated with the new parameters at the end. |
| <code>in</code>      | <code><i>coords_data</i></code>    | Pointer to single structure containing coordinates and QM energy of each input structure                                          |
| <code>in</code>      | <code><i>fit_phase</i></code>      | Whether or not phases will be fit or just dihedral force constants                                                                |

**Returns**

Integer indicating success or failure. Parm\_data is updated with the new parameters if successful

#### 4.14.3.25 int dihedral\_types\_equal ( dihedral\_type\_struct \* *first*, dihedral\_type\_struct \* *second* )

Checks if two dihedral types are the same or different. Checks with name only, not by the value of the parameters, and only checks in one direction.

**Parameters**

|    |               |                                                                  |
|----|---------------|------------------------------------------------------------------|
| in | <i>first</i>  | Pointer to first <a href="#">dihedral_type_struct</a> to examine |
| in | <i>second</i> | Pointer to <a href="#">dihedral_type_struct</a> to compare to    |

**Returns**

YES if they are the same, NO if not

#### 4.14.3.26 int dihedralcomparator ( const void \* *a*, const void \* *b* )

Compares names of two dihedrals, for qsort

#### 4.14.3.27 int do\_fit ( global\_options\_struct \* *global\_options*, parm\_struct \* *parm\_data*, coord\_set \* *coords\_data* )

Conducts the fit. Prints a parameter summary, does bounds checking, calls appropriate algorithm, verifies and prints final parameters, and writes output file formats according to the options the user has chosen.

**Parameters**

|         |                       |                                                                                                     |
|---------|-----------------------|-----------------------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure containing user defined options                                        |
| in, out | <i>parm_data</i>      | Array of initial parameters and what to fit for each molecule, will have final parameters inserted. |
| in      | <i>coords_data</i>    | Array of coordinate sets containing the atomic coordinates and QM data for input structures.        |

**Returns**

Integer indicating success or failure

#### 4.14.3.28 int do\_mutation ( global\_options\_struct \* *global\_options*, parm\_struct \* *parm\_data*, double \* *row*, int *col*, bool\_t *do\_mutate* )

Conducts mutation or a validity check on the given element in the given row of the data matrix. This is called in the genetic algorithm on an element with a set probability, most elements will never see this function.

**Parameters**

|         |                       |                                                                                       |
|---------|-----------------------|---------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                                          |
| in      | <i>parm_data</i>      | The parameter structure, used to look up what kind of parameter element to mutate is. |
| in, out | <i>row</i>            | Array representing a row in the data matrix where mutation will happen                |
| in      | <i>col</i>            | Index in the row to mutate                                                            |
| in      | <i>do_mutate</i>      | True if value is to be changed, false to just conduct a bounds check                  |

**Returns**

Integer indicating success or failure

4.14.3.29 void double\_2D\_array\_free ( double \*\* array )

4.14.3.30 int eval\_amber\_forces\_single\_struct ( global\_options\_struct \* global\_options, parm\_struct \* parm\_data, coords\_struct \* coords\_data, force\_struct \* forces, int structure )

Calculates forces on a single input structure. Uses the parameters in the parm struct and returns forces in the force struct for each atom.

**Parameters**

|         |                |                                                                            |
|---------|----------------|----------------------------------------------------------------------------|
| in      | global_options | The global options structure                                               |
| in      | parm_struct    | The parameter structure with force constants and equilibria to use         |
| in      | coords_data    | Array of coordinate structures with atom positions                         |
| in, out | forces         | Pre-allocated to array[NATOMS], will be populated with forces on each atom |
| in      | structure      | Index of the structure in coords_data to use for atom positions            |

**Returns**

Integer representing success or failure

4.14.3.31 double eval\_amber\_std\_for\_single\_struct ( global\_options\_struct \* global\_options, parm\_struct \* parm\_data, coords\_struct \* coords\_data )

Calculates the AMBER energy for a single structure.

**Parameters**

|    |                |                                                                 |
|----|----------------|-----------------------------------------------------------------|
| in | global_options | The global options structure                                    |
| in | parm_data      | Pointer to a single set of parameters to use in the calculation |
| in | coords_data    | Pointer to a single coordinate structure to evaluate            |

**Returns**

The energy of the structure

4.14.3.32 double eval\_sum\_amber\_forces ( global\_options\_struct \* global\_options, parm\_struct \* parm\_data, coord\_set \* coords\_data )

Scores a set of parameters according to force comparison. Allocates all necessary force structures, runs the force calculation (in parallel with openmp) compares the result. The comparison is the average difference in force magnitude per structure.

**Parameters**

|    |                |                                                                                    |
|----|----------------|------------------------------------------------------------------------------------|
| in | global_options | The global options structure                                                       |
| in | parm_data      | The parameter set to evaluate                                                      |
| in | coords_data    | Coordinate set containing atom coordinates for all input structures, and QM forces |

**Returns**

Scalar representing average difference in force magnitude per structure.

**4.14.3.33** `double eval_sum_amber_forces_multiprmtop ( global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_datas )`

Scores a set of parameters cross multiple molecules according to force comparison. Essentially just sums the force evaluation value over all prmtops. This is safe to use if there is only one molecule so is recommended for all force evaluation calls.

**Parameters**

|    |                       |                                                      |
|----|-----------------------|------------------------------------------------------|
| in | <i>global_options</i> | The global options structure                         |
| in | <i>parm_datas</i>     | Array of parameter sets, one for each molecule       |
| in | <i>coords_datas</i>   | Array of coordinate data sets, one for each molecule |

**Returns**

Double representing average difference in force magnitude over all molecules and structures

**4.14.3.34** `double eval_sum_squares_amber_std ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data )`

Evaluates the sum squares difference between quantum and calculated energy for each structure. The energy calculation for each structure is done in parallel with openmp.

**Parameters**

|    |                       |                                                                                                        |
|----|-----------------------|--------------------------------------------------------------------------------------------------------|
| in | <i>global_options</i> | The global options structure                                                                           |
| in | <i>parm_data</i>      | The parameter file containing parameter set to be scored                                               |
| in | <i>coords_data</i>    | Contains coordinates of atoms in all input structures, and qm energies (pointer to ONE coordinate set) |

**Returns**

The sum of the squares of the difference in energy between AMBER function evaluation and quantum value.

**4.14.3.35** `double eval_sum_squares_amber_std_multiprmtop ( global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_datas )`

Conducts the energy evaluation over every prmtop.

**Parameters**

|    |                       |                              |
|----|-----------------------|------------------------------|
| in | <i>global_options</i> | The global options structure |
| in | <i>parm_datas</i>     | The array of parm structures |
| in | <i>coords_datas</i>   | The array of coordinate sets |

**Returns**

Sum of squares of energy difference over all structures.

**4.14.3.36 void file\_open\_failure ( char \* routine, char \* var\_name )**

Prints a standard file open failure message then exits. Includes the function name and the file that was attempted to be opened.

**Parameters**

|    |                 |                                         |
|----|-----------------|-----------------------------------------|
| in | <i>routine</i>  | The function where the failure occurred |
| in | <i>var_name</i> | The filename that could not be opened   |

**4.14.3.37 int find\_atomic\_number\_from\_parm ( parm\_struct \* parm\_data, int atom )****4.14.3.38 int find\_flag ( FILE \* fptr, char \* label )**

Used to locate a title in a prmtop. Leaves the file pointer positioned at the line below the selected label. Rewinds the file pointer before searching.

**Parameters**

|         |              |                         |
|---------|--------------|-------------------------|
| in, out | <i>fptr</i>  | The file pointer to use |
| in      | <i>label</i> | The title to find       |

**Returns**

Integer SUCCESS, or INVALID\_LINE if label not found.

**4.14.3.39 int free\_coords ( global\_options\_struct \* global\_options, coord\_set \* c )**

Frees the memory used by one set of coordinates

**Parameters**

|         |                       |                                                            |
|---------|-----------------------|------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                               |
| in, out | <i>c</i>              | Pointer to the coordinate set whose contents will be freed |

**Returns**

Integer indicating success or failure

**4.14.3.40 void free\_data\_matrix ( double \*\* dm, int rows )**

Frees the data matrix used for the genetic algorithm.

**See Also**

[alloc\\_data\\_matrix](#)

**Parameters**

|         |             |                                     |
|---------|-------------|-------------------------------------|
| in, out | <i>dm</i>   | Pointer to the 2D array to be freed |
| in      | <i>rows</i> | The number of rows in the array     |

4.14.3.41 void free\_prmtop ( global\_options\_struct \* *global\_options*, parm\_struct \* *parm\_data* )

4.14.3.42 void genetic\_wizard ( global\_options\_struct \* *global\_options*, FILE \* *file* )

Reads in options specific to the genetic algorithm

#### Parameters

|         |                       |                                                |
|---------|-----------------------|------------------------------------------------|
| in, out | <i>global_options</i> | Structure where options will be set            |
| in, out | <i>file</i>           | File to save options to, or NULL if not saving |

4.14.3.43 double get\_float ( )

Gets a floating point value from stdin

#### Returns

The value that was read, as a double

4.14.3.44 int get\_option ( int *min*, int *max* )

Gets an integer in the specified range from stdin

#### Parameters

|    |            |                                      |
|----|------------|--------------------------------------|
| in | <i>min</i> | The minimum allowed value, exclusive |
| in | <i>max</i> | The maximum allowed value, exclusive |

#### Returns

The integer that was read

4.14.3.45 void global\_unlock ( global\_options\_struct \* *global\_options*, parm\_struct \*\* *parm\_data*, coord\_set \*\* *coords\_data* )

4.14.3.46 void handle\_sigint ( int *param* )

Prints an error message on a signal. This may someday print something more useful.

#### Parameters

|    |               |                           |
|----|---------------|---------------------------|
| in | <i>signal</i> | The signal that is caught |
|----|---------------|---------------------------|

4.14.3.47 int job\_control\_wizard ( global\_options\_struct \* *global\_options* )

The main job control wizard. Walks you through the various options and lets you save them as you go.

#### Parameters

|         |                       |                                                      |
|---------|-----------------------|------------------------------------------------------|
| in, out | <i>global_options</i> | Will contain options set by wizard, file to write to |
|---------|-----------------------|------------------------------------------------------|



## Returns

Integer indicating success or failure.

### 4.14.3.48 void malloc\_failure\_char ( char \* *routine*, char \* *var\_name*, int *chars\_requested* )

Prints a standard malloc failure message for char data types then exits. Includes the routine name, variable name, and number of bytes requested for char data types

#### Parameters

|    |                        |                                                    |
|----|------------------------|----------------------------------------------------|
| in | <i>routine</i>         | The function where the failure occurred            |
| in | <i>var_name</i>        | The variable that failed to be allocated           |
| in | <i>chars_requested</i> | The number of characters requested to be allocated |

### 4.14.3.49 void malloc\_failure\_double ( char \* *routine*, char \* *var\_name*, int *doubles\_requested* )

Prints out a standard malloc failure message for double data types then exits. Includes the routine name, variable name, and number of bytes requested for double data types.

#### Parameters

|    |                             |                                                 |
|----|-----------------------------|-------------------------------------------------|
| in | <i>routine</i>              | The function where the failure occurred         |
| in | <i>var_name</i>             | The variable that failed to be allocated        |
| in | <i>doubles_ - requested</i> | The number of doubles that were to be allocated |

### 4.14.3.50 void malloc\_failure\_int ( char \* *routine*, char \* *var\_name*, int *ints\_requested* )

Prints out a standard malloc failure message for int data types then exits. Includes the routine name, variable name, and number of bytes requested for int data types.

#### Parameters

|    |                       |                                                  |
|----|-----------------------|--------------------------------------------------|
| in | <i>routine</i>        | The function where the failure occurred          |
| in | <i>var_name</i>       | The variable that failed to be allocated         |
| in | <i>ints_requested</i> | The number of integers that were to be allocated |

### 4.14.3.51 void malloc\_failure\_short\_int ( char \* *routine*, char \* *var\_name*, int *short\_ints\_requested* )

Prints out a standard malloc failure message for short int data types then exits. Includes the routine name, variable name, and number of bytes requested for short int data types.

#### Parameters

|    |                                |                                                    |
|----|--------------------------------|----------------------------------------------------|
| in | <i>routine</i>                 | The function where the failure occurred            |
| in | <i>var_name</i>                | The variable that failed to be allocated           |
| in | <i>short_ints_ - requested</i> | The number of short ints that were to be allocated |

**4.14.3.52** `int minimise_function_genetic ( global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_data )`

Conducts the majority of the genetic algorithm minimization.

#### Parameters

|         |                       |                                                                                     |
|---------|-----------------------|-------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure, with algorithm options inside                         |
| in, out | <i>parm_datas</i>     | Array containing the parameters for each molecule, will be updated each generation  |
| in      | <i>coords_data</i>    | Array containing coordinates and QM data for each input structure for each molecule |

#### Returns

Integer indicating success or failure.

**4.14.3.53** `int minimise_function_simplex ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data )`

The simplex minimization function. Written mostly by Ross Walker

#### Parameters

|         |                       |                                                                           |
|---------|-----------------------|---------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                              |
| in, out | <i>parm_data</i>      | Array of parameter structures, updated at each iteration                  |
| in      | <i>coords_data</i>    | Array of coordinate sets containing all the input structures with QM data |

#### Returns

Integer indicating success or failure

**4.14.3.54** `int modify_params_scratch_data ( global_options_struct * global_options, parm_struct * parm_data, double * parameters, readwrite_t MODE )`

Reads and writes parameters in the [parm\\_struct](#) data structure. Extracts parameters marked as variable from the prmtop and puts them in the linear array, or takes parameters from the array and puts them in the data structure. The order in which the extraction done is as follows: K BOND x (KR, KEQ) BOND y (KR, KEQ) ANGLE x (KT, THEQ) ANGLE y (KT, THEQ) DIHEDRAL x (KP, PN, PHASE) DIHEDRAL y (KP, PN, PHASE)

#### Parameters

|         |                       |                                                                                                  |
|---------|-----------------------|--------------------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                                                     |
| in, out | <i>parm_data</i>      | The parameters data structure to read or write to                                                |
| in, out | <i>parameters</i>     | Pre-allocated array size NDIMENSIONS to read or write to                                         |
| in      | <i>MODE</i>           | if READ, copy the data from parm_data to parameters. if WRITE, copy from parameter to parm_data. |

#### Returns

The number of parameters successfully extracted.

I am aware that this whole procedure here is clunky and slow but it makes it significantly easier to understand and debug. At some point I will replace this whole system with a much more efficient method

4.14.3.55 `int name_copy ( FILE * fptr, char * stringp )`

Desigend to read 4 characters from a file stream and null terminate the string.

See Also

[prmtop\\_params.h](#) for definition of `NAME_SIZE`

## Parameters

|                      |                             |                           |
|----------------------|-----------------------------|---------------------------|
| <code>in, out</code> | <code><i>fptr</i></code>    | File pointer to read from |
| <code>in, out</code> | <code><i>stringp</i></code> | The string to read into   |

## Returns

Integer indicating SUCCESS or INVALID\_LINE

4.14.3.56 `int not_enough_dihedrals ( parm_struct * parm_data, int n )`

Potentially adds more dihedral terms to existing ones so you can get a better fit. This is completely deprecated by dihedral data refactoring and so is commented out for now.

4.14.3.57 `int ObfuscateAtom ( int at )`

Re-obfuscates an atom from atomic number into prmtop format. This can be used in writing a prmtop.

See Also

[unObfuscateAtom](#)

## Parameters

|                 |                        |                       |
|-----------------|------------------------|-----------------------|
| <code>in</code> | <code><i>at</i></code> | The atom to obfuscate |
|-----------------|------------------------|-----------------------|

## Returns

The prmtop obfuscated atom number.

4.14.3.58 `void print_atomic_number_as_symbol ( FILE * fptr, int atomic_number )`4.14.3.59 `void print_backtrace ( int signal )`

Prints a backtrace for debugging. This does not work in Cygwin and so is ifdef'd out in that case.

## Parameters

|                 |                            |                           |
|-----------------|----------------------------|---------------------------|
| <code>in</code> | <code><i>signal</i></code> | The signal that is caught |
|-----------------|----------------------------|---------------------------|

4.14.3.60 `void print_close_line_box ( int no_spaces )`

Prints the closing of a line in a box styled like

||

## See Also

[print\\_open\\_line\\_box](#)

## Parameters

|                 |                        |                                            |
|-----------------|------------------------|--------------------------------------------|
| <code>in</code> | <code>no_spaces</code> | The number of spaces to go before the line |
|-----------------|------------------------|--------------------------------------------|

4.14.3.61 `void print_dihedral ( dihedral_type_struct * type, int term )`

Pretty prints one dihedral value and parameters

## Parameters

|                 |                   |                                                                  |
|-----------------|-------------------|------------------------------------------------------------------|
| <code>in</code> | <code>type</code> | Pointer to the <a href="#">dihedral_type_struct</a> to print     |
| <code>in</code> | <code>term</code> | Which term of this dihedral the parameters should be printed for |

4.14.3.62 `void print_forces ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data, int atom )`

Prints out a nice table with forces in amber and quantum for one atom over each structure. It will conduct the forces evaluation itself, including allocation of force structures.

## Parameters

|                 |                             |                                                              |
|-----------------|-----------------------------|--------------------------------------------------------------|
| <code>in</code> | <code>global_options</code> | The global options structure                                 |
| <code>in</code> | <code>parm_data</code>      | The parameter set to use in the function                     |
| <code>in</code> | <code>coords_data</code>    | Pointer to single set of atom coordinates in input structure |
| <code>in</code> | <code>atom</code>           | Index of the atom to print out. 0 is usually a good choice.  |

4.14.3.63 `void print_job_control_summary ( global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_datas )`

Prints a summary of the options in the job control file

## Parameters

|                 |                             |                                              |
|-----------------|-----------------------------|----------------------------------------------|
| <code>in</code> | <code>global_options</code> | The global options structure                 |
| <code>in</code> | <code>parm_data</code>      | Array of parameter files                     |
| <code>in</code> | <code>coords_datas</code>   | Array of coordinate sets of input structures |

4.14.3.64 `void print_multiprmtop_summary ( global_options_struct * global_options, parm_struct * parm_datas )`

Prints out a summary of only the parameters to be fit over a set of multiple prmtops. Assumes that each prmtop contains every single parameter to be fit, because only the data from the first one is used.

## See Also

[verify\\_prmtops](#)

## Parameters

|    |                       |                               |
|----|-----------------------|-------------------------------|
| in | <i>global_options</i> | The global options structure  |
| in | <i>parm_dats</i>      | Array of parameter structures |

## 4.14.3.65 void print\_open\_line\_box ( int \* i )

Prints the beginning of an ASCII box.

## See Also

[print\\_close\\_line\\_box](#)

## Parameters

|         |          |                                       |
|---------|----------|---------------------------------------|
| in, out | <i>i</i> | Contains number of characters printed |
|---------|----------|---------------------------------------|

## 4.14.3.66 void print\_parameter\_summary ( global\_options\_struct \* global\_options, parm\_struct \* parm\_data )

Prints out a summary of all parameters currently stored in one parm\_data. A \* will mark which parameters are being fit.

## Parameters

|                       |                                  |
|-----------------------|----------------------------------|
| <i>global_options</i> | The global options structure     |
| <i>parm_data</i>      | The parameter structure to print |

## 4.14.3.67 void print\_program\_history ( void )

## 4.14.3.68 void print\_program\_info ( void )

## 4.14.3.69 int process\_command\_line ( int argc, char \*\* argv, global\_options\_struct \* global\_options, parm\_struct \*\* parm\_dats, coord\_set \*\* coords\_dats )

Processes all command line options. Possible options combinations are: paramfit -i [Job control file] -p [prmtop] -c [mdcrd] -q [quantum data] -d [ON/OFF/DEBUG] --random-seed [seed] paramfit -i [Job control file] -pf [prmtop list] -cf [mdcrd list] -qf [quantum list] -d [ON/OFF/DEBUG] --random-seed [seed]

## Parameters

|         |                       |                                                                                           |
|---------|-----------------------|-------------------------------------------------------------------------------------------|
| in      | <i>argv</i>           | Pointer to array with all options, argv[0] is the name of the program (paramfit)          |
| in, out | <i>global_options</i> | Global options structure that will be updated                                             |
| in, out | <i>parm_dats</i>      | Unallocated array of parm structs, will be updated with initial blank one if -p specified |
| in, out | <i>coords_dats</i>    | Unallocated of coords structs, will be updated with initial blank one if -c specified     |

**Returns**

Integer indicating success, failure, options problems, etc.

**4.14.3.70** `int process_job_control_setting ( char * setting_line, int length, int * number_settings, global_options_struct * global_options, coord_set * coords_data )`

Processes one line of the job control file and sets corresponding options. This function is called repeatedly for each line of the job control file until all settings are processed.

**See Also**

[read\\_job\\_control\\_file](#)

**Parameters**

|         |                        |                                                                                                                                                                                      |
|---------|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in      | <i>setting_line</i>    | The raw line from the job control file as a character array                                                                                                                          |
| in      | <i>length</i>          | The length of this line                                                                                                                                                              |
| in, out | <i>number_settings</i> | The number of settings that have been set. Will be updated if successful.                                                                                                            |
| in, out | <i>global_options</i>  | The global options structure to input settings to                                                                                                                                    |
| in, out | <i>coords_data</i>     | Pointer to array of coords struct already allocated to size 1 that may have its number of structures set if NSTRUCTURES is specified in the job control file, for single prmtop fits |

**Returns**

Integer indicating success or various types of failure (invalid setting/data, alloc fail, etc)

**4.14.3.71** `int process_prmtops ( global_options_struct * global_options, parm_struct * parm_dats )`

Processes all the parmtop data. Processes each individual [parm\\_struct](#).

**Parameters**

|         |                       |                                                           |
|---------|-----------------------|-----------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure, specifies number of prmtops |
| in, out | <i>parm_dats</i>      | Array of parm structures to process                       |

**Returns**

Integer indicating success or failure

**4.14.3.72** `void process_retval ( int err_code, verbosity_t VERBOSITY )`

Processes the return value of a function and exits if an error

**Parameters**

|    |                  |                                    |
|----|------------------|------------------------------------|
| in | <i>err_code</i>  | The return value from the function |
| in | <i>VERBOSITY</i> | How verbose the program is         |

4.14.3.73 int process\_single\_prmtop ( global\_options\_struct \* *global\_options*, parm\_struct \* *parm\_data* )

Processes raw prmtop data into arrays of unique bond, angle, and dihedral parameters that are in structures that are much easier to optimize. Done on one prmtop at a time

## See Also

[read\\_prmtops](#)  
[process\\_prmtops](#)

## Parameters

|         |                       |                                                                                     |
|---------|-----------------------|-------------------------------------------------------------------------------------|
| in, out | <i>global_options</i> | The global options structure. unique_[bonds,angles,dihedrals]_found will be updated |
| in, out | <i>parm_data</i>      | Contains the raw prmtop data, will also be populated with processed prmtop data     |

## Returns

Integer indicating success or failure

4.14.3.74 int read\_gaussian\_energy ( global\_options\_struct \* *global\_options*, parm\_struct \* *parm\_data*, coord\_set \* *coords\_data* )

Reads in the gaussian energy from a certain set of files into a coordinate structure This lets paramfit write the file, run gaussian, read the file without any need for the user to go through the gaussian output.

## Parameters

|         |                       |                                                    |
|---------|-----------------------|----------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                       |
| in      | <i>parm_data</i>      | Pointer to single parm struct                      |
| in, out | <i>coords_data</i>    | Pointer to single coord struct to attach energy to |

## Returns

Integer indicating success or failure

4.14.3.75 int read\_gaussian\_forces ( global\_options\_struct \* *global\_options*, parm\_struct \* *parm\_data*, coord\_set \* *coords\_data* )

Reads in Gaussian output files with forces for all structures. The files need to have been generated in the CREATE\_INPUT mode so that the atom numbering is consistent. This also lets us assume that the files are named QMFILEOUTSTART###QMFILEOUTEND.out

## Parameters

|         |                       |                                                                    |
|---------|-----------------------|--------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                       |
| in      | <i>parm_data</i>      | Pointer to single parameter file                                   |
| in, out | <i>coords_data</i>    | Pointer to single coordinate set, will have forces attached to it. |

## Returns

Integer indicating success or failure.

4.14.3.76 `int read_input_parameters ( global_options_struct * global_options, parm_struct * parm_data )`

4.14.3.77 `int read_job_control_file ( global_options_struct * global_options, coord_set * coords_data )`

Reads the job control file and puts the options into the options structure.

#### Parameters

|         |                       |                                                                                           |
|---------|-----------------------|-------------------------------------------------------------------------------------------|
| in, out | <i>global_options</i> | The global options structure, which contains the location of the job control file.        |
| in, out | <i>coords_data</i>    | Pointer to array of coords struct already allocated to size 1 that may have its number of |

#### Returns

Integer indicating success or failure

4.14.3.78 `int read_mdcrds ( global_options_struct * global_options, parm_struct * parm_dats, coord_set ** s_dats )`

Reads in each coordinate file from the list of coordinate file.

#### Parameters

|     |                       |                                                                                                                                                                         |
|-----|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in  | <i>global_options</i> | The global options structure                                                                                                                                            |
| in  | <i>parm_dats</i>      | The array of parameter sets to use                                                                                                                                      |
| out | <i>struc_dats</i>     | Pointer to array of structure sets to be initialized. The array is size 1 already if -c command line option for single mdcrd has been specified, otherwise *s_dats=NULL |

#### Returns

Integer indicating success or failure

4.14.3.79 `int read_parameter_file ( global_options_struct * global_options, parm_struct * parm_data )`

Reads in a previously saved list of parameters to fit. The parameters are in the exact order as the prmtop, making this non-transferable between prmtops. This is now included for backwards-compatibility. It will check if you have a nwe format parameter file and call the v2 function to read it if found.

#### See Also

[read\\_parameter\\_file\\_v2](#)

#### Parameters

|         |                       |                                                                  |
|---------|-----------------------|------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure, containing the file name           |
| in, out | <i>parm_data</i>      | The parameter data file, will be updated with parameters to fit. |



**Returns**

Integer indicating success or failure.

#### 4.14.3.80 `int read_prmtops ( global_options_struct * global_options, parm_struct ** parm_dats )`

Reads in all prmtop data. Starts by getting filenames one line at a time from the prmtop list and then reading in data from each one of those.

**See Also**

[read\\_single\\_prmtop](#)

**Parameters**

|         |                       |                                                           |
|---------|-----------------------|-----------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure, with the parm list filename |
| in, out | <i>parm_data</i>      | Pointer to an array of parm_structs, will be reallocd.    |

**Returns**

Integer indicating success or failure

#### 4.14.3.81 `int read_qm ( global_options_struct * global_options, parm_struct * parm_dats, coord_set * coords_dats )`

The master function for all reading of QM input data. Will call functions to read in either forces or energies from a list or from raw output files from the appropriate program.

**Parameters**

|         |                       |                                                                                                                                            |
|---------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure, which specifies the data format and energy or forces                                                         |
| in      | <i>parm_dats</i>      | Array of parameter sets corresponding to coordinates to read in                                                                            |
| in, out | <i>coords_dats</i>    | Array of coordinate sets that each have the filename/folder to read from initialized. These will be updated with the QM energies or forces |

**Returns**

Integer indicating success or failure

#### 4.14.3.82 `int read_qm_directory ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data )`

Reads all qm output files in a given directory. The names of the files are according to the convention paramfit writes them in with CREATE\_INPUT mode- that is prmtop.QMFILEOUTEND.## where prmtop is prmtop->filename. This will support a variety of file formats, which one it is is set in global\_options->QMFORMAT or similar.

**Parameters**

|         |                       |                                                                                                                                          |
|---------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                                                                                             |
| in, out | <i>coords_data</i>    | Pointer to single coordinate set. Each structure in the set will be populated with the energy / forces from the matching QM output file. |

**Returns**

Integer indicating success or failure

**4.14.3.83** `int read_qm_energy_list ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data )`

Reads in the quantum energies and attaches one to each coordinate structure. Performs unit conversion if necessary. This is used when evaluating energies, NOT forces. The QM data file should consist of NSTRUCTURES worth of doubles, with one value per line.

**Parameters**

|                      |                             |                                                                            |
|----------------------|-----------------------------|----------------------------------------------------------------------------|
| <code>in</code>      | <code>global_options</code> | The global options structure, containing filename with energies            |
| <code>in</code>      | <code>parm_data</code>      | Pointer to single parameter structure corresponding to this coordinate set |
| <code>in, out</code> | <code>coords_data</code>    | Pointer to single coordinate set that energies will be attached to.        |

**Returns**

Integer indicating success or failure.

**4.14.3.84** `int read_single_mdcrd ( coord_set * coords_data )`

Reads the coordinate file with all the input structures.

**Parameters**

|                      |                          |                                                                                                                                                                                         |
|----------------------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>in, out</code> | <code>coords_data</code> | Pointer to coordinate set structure that will be populated. Filename and number of structures needs to be already initialized. The coordinate data set should also have been allocated. |
|----------------------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**See Also**

[alloc\\_coords](#)  
[read\\_mdcrds](#)

**Returns**

Integer indicating success or failure.

**4.14.3.85** `int read_single_prmtop ( global_options_struct * global_options, parm_struct * parm_data )`

Reads the raw data from the prmtop file. Sorts out how many atoms, atom types, and basic info. Does not do any processing into the fitting data structure.

**See Also**

[process\\_prmtop.c](#)

**Parameters**

|                      |                                     |                                                    |
|----------------------|-------------------------------------|----------------------------------------------------|
| <code>in</code>      | <code>global_options</code>         | The global options structure                       |
| <code>in, out</code> | <code>parm_data</code>              | The parameter structure that will have info put in |
| <code>in</code>      | <code>parm_data-&gt;filename</code> | The path to the prmtop file to read                |

4.14.3.86 int s\_getline ( char \* *line*, int *max*, FILE \* *fp* )

"Safe" version of getline, exits with an error if EOF is found.

## Parameters

|         |             |                                              |
|---------|-------------|----------------------------------------------|
| in, out | <i>line</i> | String that will contain the read line       |
| in      | <i>max</i>  | The maximum number of characters to read     |
| in      | <i>fp</i>   | File pointer to the file to read a line from |

## Returns

The number of characters read, or FILE\_READ\_FAIL if EOF found

4.14.3.87 int set\_default\_options ( global\_options\_struct \* *global\_options* )

Initializes variables representing program options to their defaults.

## Parameters

|         |                       |                                                          |
|---------|-----------------------|----------------------------------------------------------|
| in, out | <i>global_options</i> | The global options structure where defaults will be set. |
|---------|-----------------------|----------------------------------------------------------|

## Returns

Integer indicating success or failure.

4.14.3.88 void set\_dihedral\_fit ( global\_options\_struct \* *global\_options*, dihedral\_type\_struct \* *s*, int *t*, bool\_t *d\_kp*, bool\_t *d\_np*, bool\_t *d\_phase* )

Sets dihedral fitting options for a given dihedral. All parameters for this term should have already been initialized.

## Parameters

|         |                       |                                                     |
|---------|-----------------------|-----------------------------------------------------|
| in      | <i>global_options</i> | The global options structure with prompting options |
| in, out | <i>s</i>              | The dihedral type to set options for                |
| in      | <i>t</i>              | Integer indicating the term to fit                  |
| in      | <i>d_kp</i>           | Whether to prompt to fit dihedral KP                |
| in      | <i>d_np</i>           | Whether to prompt to fit dihedral NP                |
| in      | <i>d_phase</i>        | Whether to prompt to fit dihedral PHASE             |

4.14.3.89 void simplex\_wizard ( global\_options\_struct \* *global\_options*, FILE \* *file* )

Reads in options specific to the simplex algorithm

## Parameters

|         |                       |                                                |
|---------|-----------------------|------------------------------------------------|
| in, out | <i>global_options</i> | Structure where options will be set            |
| in, out | <i>file</i>           | File to save options to, or NULL if not saving |

4.14.3.90 int unObfuscateAtom ( int *at* )

Translates atom numbers from prmtop format into understandable numbers. PARM for some reason obfuscates each atom number by (when positive),  $at = (at+3) / 3$ . This function returns the true atom number. For example,

unObfuscateAtom(24) = 9.

#### See Also

[ObfuscateAtom](#)

#### Parameters

|           |           |                                  |
|-----------|-----------|----------------------------------|
| <i>in</i> | <i>at</i> | The obfuscated atom to translate |
|-----------|-----------|----------------------------------|

#### Returns

The true atom number

4.14.3.91 `int update_prmtop_data ( global_options_struct * global_options, parm_struct * parm_dats, double * parameters )`

Puts the parameters in a data array into each prmtop. Used in multi-prmtop fits to update each of them before conducting an energy evaluation.

#### Parameters

|                |                       |                                                          |
|----------------|-----------------------|----------------------------------------------------------|
| <i>in</i>      | <i>global_options</i> | The global options structure                             |
| <i>in, out</i> | <i>parm_data</i>      | Array of parameter data structures that will be updated. |
| <i>in</i>      | <i>parameters</i>     | Pre-allocated array size NDIMENSIONS to read from        |

#### Returns

Integer indicating success or failure.

4.14.3.92 `int verify_prmtops ( global_options_struct * global_options, parm_struct * parm_dats )`

Checks that each prmtop contains all of the parameters that are to be fit. This is necessary for consistency in the fitting, as there is no point in doing a multi-molecule fit if a parameter is only present in one of the molecules.

#### Parameters

|           |                       |                              |
|-----------|-----------------------|------------------------------|
| <i>in</i> | <i>global_options</i> | The global options structure |
| <i>in</i> | <i>parm_dats</i>      | Array of parameter structure |

#### Returns

Integer indicating success (prmtops okay) or failure (inconsistency detected or other error)

4.14.3.93 `int write_energy ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data, int generation )`

Writes a data file with amber and quantum energies for each structure with given parameters. Useful to plot the results of a calculation to see the quality of fit.

#### Parameters

|           |                       |                                                                        |
|-----------|-----------------------|------------------------------------------------------------------------|
| <i>in</i> | <i>global_options</i> | The global options structure, with filename to save to                 |
| <i>in</i> | <i>parm_data</i>      | Pointer to single parameter set to use when calculating amber energies |
| <i>in</i> | <i>coords_data</i>    | Pointer to single coordinate set with attached quantum energy          |
| <i>in</i> | <i>generation</i>     | The generation of the genetic algorithm, so can be called iteratively  |

## Returns

Integer indicating success or failure

4.14.3.94 `int write_frcmod ( global_options_struct * global_options, parm_struct * parm_data )`

Write a force field modification file. To be used to save the results of a fit for reading into Leap

## Parameters

|    |                       |                                                            |
|----|-----------------------|------------------------------------------------------------|
| in | <i>global_options</i> | The global options structure, containing filename to write |
| in | <i>parm_data</i>      | The parameter file containing fitted results               |

## Returns

Integer indicating success or failure

4.14.3.95 `int write_input_adf ( global_options_struct * global_options, parm_struct * parm_data, coord_set * current_struct, int num, FILE * fptr )`

Writes an input file for ADF for a structure. The file is initialized in write quantum. Note that this file is only for energy calculations, not forces.

## Parameters

|         |                       |                                                                     |
|---------|-----------------------|---------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                        |
| in      | <i>parm_data</i>      | Pointer to the parameter structure that describes these coordinates |
| in      | <i>current_struct</i> | Pointer to the coordinate set that will be used                     |
| in      | <i>num</i>            | Number of the structure in teh coordinate set to write              |
| in, out | <i>fptr</i>           | Pointer to the file to write to, initialized elsewhere              |

## Returns

Integer indicating success or failure

4.14.3.96 `int write_input_gamess ( global_options_struct * global_options, parm_struct * parm_data, coord_set * current_struct, int num, FILE * fptr )`

Writes a quantum input file for GAMESS. This is for energy calculation only.

## Parameters

|         |                          |                                                                        |
|---------|--------------------------|------------------------------------------------------------------------|
| in      | <i>global_options</i>    | The global options structure                                           |
| in      | <i>parm_data</i>         | The parameter structure                                                |
| in      | <i>current_structure</i> | Pointer to coordinate set to write from                                |
| in      | <i>num</i>               | The number of the structure in the coordinate set that will be written |
| in, out | <i>fptr</i>              | File to write to, should be already initialized                        |

## Returns

Integer indicating success or failure.

**4.14.3.97** `int write_input_gaussian ( global_options_struct * global_options, parm_struct * parm_data, coord_set * current_struct, int num, FILE * fptr )`

Writes a gaussian input file for one structure. Should not be called directly. Uses a header file containing all of the quantum options that the atom coordinates are appended to, so this can be used to make input files for force or energy calculations.

#### See Also

`write_input`

#### Parameters

|         |                       |                                                              |
|---------|-----------------------|--------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                 |
| in      | <i>parm_data</i>      | The parameter file                                           |
| in      | <i>current_struct</i> | Pointer to coordinate set to use                             |
| in      | <i>num</i>            | The number of the structure to write from the coordinate set |
| in, out | <i>fptr</i>           | Pointer to the file to write to                              |

#### Returns

Integer indicating success or failure

**4.14.3.98** `int write_input_parameters ( global_options_struct * global_options, parm_struct * parm_data )`

Writes a file of which parameters are to be fit. This can be read in later to fit multiple runs with one set of parameters to fit. Assumes if multiple prmtops, the parameters are the same in each one

#### Parameters

|    |                       |                                                              |
|----|-----------------------|--------------------------------------------------------------|
| in | <i>global_options</i> | Global options structure, containing filename to save ase    |
| in | <i>parm_data</i>      | The parameter file containing which parameters are to be fit |

#### Returns

Integer indicating success or failure

**4.14.3.99** `int write_prmtop ( global_options_struct * global_options, parm_struct * parm_data )`

Saves a prmtop with the given parameters. Really experimental, and I'm not sure if this is even useful.

#### Parameters

|    |                       |                                                             |
|----|-----------------------|-------------------------------------------------------------|
| in | <i>global_options</i> | The global options structure, including the file to save to |
| in | <i>parm_data</i>      | The parameters to put into the prmtop file                  |

**Returns**

Integer indicating success or failure

## 4.15 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/genetic\_algorithm.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "function_def.h"
#include "constants.h"
```

**Functions**

- int [minimise\\_function\\_genetic](#) ([global\\_options\\_struct](#) \*global\_options, [parm\\_struct](#) \*parm\_datas, [coord\\_set](#) \*coords\_data)
- int [do\\_mutation](#) ([global\\_options\\_struct](#) \*global\_options, [parm\\_struct](#) \*parm\_data, double \*row, int col, [bool\\_t](#) do\_mutate)
- double \*\* [alloc\\_data\\_matrix](#) (int rows, int cols)
- void [free\\_data\\_matrix](#) (double \*\*dm, int rows)

**4.15.1 Detailed Description**

Contains functions used by the genetic algorithm fitting.

**4.15.2 Function Documentation****4.15.2.1 double\*\* alloc\_data\_matrix ( int rows, int cols )**

Allocates a more traditional, non-contiguous 2D array for the genetic algorithm. We don't use `alloc_2d_double` because that gets a contiguous block of memory and we specifically need non-contiguous since we will be swapping the rows around in sorting each generation, and free-ing the contiguous matrix later is a pain since it is now unclear which was the original first row.

**See Also**

[free\\_data\\_matrix](#)

**Parameters**

|    |             |                                          |
|----|-------------|------------------------------------------|
| in | <i>rows</i> | The number of rows to be in the array    |
| in | <i>cols</i> | The number of columns to be in the array |

**Returns**

Pointer to a 2D double array of the specified size

**4.15.2.2** `int do_mutation ( global_options_struct * global_options, parm_struct * parm_data, double * row, int col, bool_t do_mutate )`

Conducts mutation or a validity check on the given element in the given row of the data matrix. This is called in the genetic algorithm on an element with a set probability, most elements will never see this function.

**Parameters**

|         |                       |                                                                                       |
|---------|-----------------------|---------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                                          |
| in      | <i>parm_data</i>      | The parameter structure, used to look up what kind of parameter element to mutate is. |
| in, out | <i>row</i>            | Array representing a row in the data matrix where mutation will happen                |
| in      | <i>col</i>            | Index in the row to mutate                                                            |
| in      | <i>do_mutate</i>      | True if value is to be changed, false to just conduct a bounds check                  |

**Returns**

Integer indicating success or failure

**4.15.2.3** `void free_data_matrix ( double ** dm, int rows )`

Frees the data matrix used for the genetic algorithm.

**See Also**

[alloc\\_data\\_matrix](#)

**Parameters**

|         |             |                                     |
|---------|-------------|-------------------------------------|
| in, out | <i>dm</i>   | Pointer to the 2D array to be freed |
| in      | <i>rows</i> | The number of rows in the array     |

**4.15.2.4** `int minimise_function_genetic ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data )`

Conducts the majority of the genetic algorithm minimization.

**Parameters**

|         |                       |                                                                                     |
|---------|-----------------------|-------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure, with algorithm options inside                         |
| in, out | <i>parm_data</i>      | Array containing the parameters for each molecule, will be updated each generation  |
| in      | <i>coords_data</i>    | Array containing coordinates and QM data for each input structure for each molecule |



## Returns

Integer indicating success or failure.

## 4.16 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/help\_functions.c File Reference

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

## Functions

- void [command\\_line\\_help](#) (char \*cmd\_line\_options[])

### 4.16.1 Function Documentation

4.16.1.1 void [command\\_line\\_help](#) ( char \* *cmd\_line\_options* [ ] )

## 4.17 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/mem\_alloc.c File Reference

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

## Functions

- double \*\* [alloc\\_2D\\_double](#) (int nrows, int ncolumns)
- void [double\\_2D\\_array\\_free](#) (double \*\*array)
- void [global\\_unlock](#) ([global\\_options\\_struct](#) \*global\_options, [parm\\_struct](#) \*\*parm\_datas, [coord\\_set](#) \*\*coords\_data)
- void [free\\_prmtop](#) ([global\\_options\\_struct](#) \*global\_options, [parm\\_struct](#) \*parm\_data)
- int [alloc\\_coords](#) ([global\\_options\\_struct](#) \*global\_options, [coord\\_set](#) \*c)
- int [free\\_coords](#) ([global\\_options\\_struct](#) \*global\_options, [coord\\_set](#) \*c)

### 4.17.1 Function Documentation

4.17.1.1 double\*\* [alloc\\_2D\\_double](#) ( int *nrows*, int *ncolumns* )

4.17.1.2 int [alloc\\_coords](#) ( [global\\_options\\_struct](#) \* *global\_options*, [coord\\_set](#) \* *c* )

Allocates memory for one set of coordinates. Allocates a structure for each input conformation, and for each of those allocates space for the x,y,z coordinates of each atom.

## Parameters

|         |                       |                                                                                           |
|---------|-----------------------|-------------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                                              |
| in, out | <i>c</i>              | Pointer to coordinate set to allocate for, with filename,natoms and nstructures specified |

**Returns**

Integer indicating success or failure

4.17.1.3 `void double_2D_array_free ( double ** array )`

4.17.1.4 `int free_coords ( global_options_struct * global_options, coord_set * c )`

Frees the memory used by one set of coordinates

**Parameters**

|         |                       |                                                            |
|---------|-----------------------|------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                               |
| in, out | c                     | Pointer to the coordinate set whose contents will be freed |

**Returns**

Integer indicating success or failure

4.17.1.5 `void free_prm_top ( global_options_struct * global_options, parm_struct * parm_data )`

4.17.1.6 `void global_unlock ( global_options_struct * global_options, parm_struct ** parm_datas, coord_set ** coords_data )`

## 4.18 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/misc\_utils.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include "constants.h"
#include "function_def.h"
```

**Functions**

- void [print\\_close\\_line\\_box](#) (int no\_spaces)
- void [print\\_open\\_line\\_box](#) (int \*i)
- int [check\\_for\\_valid\\_filename](#) (const char \*data\_string, const int length)
- int [s\\_getline](#) (char \*line, int max, FILE \*fp)
- int [find\\_flag](#) (FILE \*fp, char \*label)
- int [name\\_copy](#) (FILE \*fp, char \*stringp)
- int [unObfuscateAtom](#) (int at)
- int [ObfuscateAtom](#) (int at)
- double [calc\\_bond\\_length](#) (double bond1x, double bond1y, double bond1z, double bond2x, double bond2y, double bond2z)
- double [calc\\_angle\\_radians](#) (double atom1x, double atom1y, double atom1z, double atom2x, double atom2y, double atom2z, double atom3x, double atom3y, double atom3z)
- double [calc\\_dihedral\\_radians](#) (double atom1x, double atom1y, double atom1z, double atom2x, double atom2y, double atom2z, double atom3x, double atom3y, double atom3z, double atom4x, double atom4y, double atom4z)
- void [calc\\_fit\\_dimensions](#) (global\_options\_struct \*global\_options, parm\_struct \*parm\_datas)
- int [update\\_prm\\_top\\_data](#) (global\_options\_struct \*global\_options, parm\_struct \*parm\_datas, double \*parameters)

- int `modify_params_scratch_data` (`global_options_struct` \*global\_options, `parm_struct` \*parm\_data, double \*parameters, `readwrite_t` MODE)
- void `print_backtrace` (int signal)
- void `handle_sigint` (int param)
- int `calculate_no_fit_params` (`parm_struct` \*parm\_data, short int MODE)
- int `dihedral_types_equal` (`dihedral_type_struct` \*first, `dihedral_type_struct` \*second)
- void `print_dihedral` (`dihedral_type_struct` \*type, int term)
- int `compare_energy` (const void \*a, const void \*b)
- int `not_enough_dihedrals` (`parm_struct` \*parm\_data, int n)

### 4.18.1 Detailed Description

Contains a number of small utilities used in the program that don't really have anywhere else to go.

### 4.18.2 Function Documentation

**4.18.2.1** `double calc_angle_radians ( double atom1x, double atom1y, double atom1z, double atom2x, double atom2y, double atom2z, double atom3x, double atom3y, double atom3z )`

Calculates the angle between 3 3-dimensional points Angle between  $\vec{a}_1, \vec{a}_2, \vec{a}_3$  is defined as:  $\vec{v}_1 = \vec{a}_1 - \vec{a}_2$   $\vec{v}_2 = \vec{a}_3 - \vec{a}_2$

$$\theta = \cos^{-1} \frac{\vec{v}_1 \cdot \vec{v}_2}{|\vec{v}_1| |\vec{v}_2|}$$

Returns

The angle between the three atoms

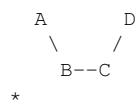
**4.18.2.2** `double calc_bond_length ( double bond1x, double bond1y, double bond1z, double bond2x, double bond2y, double bond2z )`

Calculates the distance between 2 3-dimensional points.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

**4.18.2.3** `double calc_dihedral_radians ( double atom1x, double atom1y, double atom1z, double atom2x, double atom2y, double atom2z, double atom3x, double atom3y, double atom3z, double atom4x, double atom4y, double atom4z )`

Calculates the dihedral between 4 points in 3 dimensions Given four atoms A,B,C & D:



The torsion angle along the torsion axis B-C is the angle between the planes ABC and BCD. The best way to calculate this is in terms of 3 vectors:

$a = A \rightarrow B$   $b = B \rightarrow C$   $c = C \rightarrow D$

In terms of these vectors the dihedral angle is then:  $\theta = \cos^{-1} \frac{(\vec{a} \times \vec{b})(\vec{b} \times \vec{c})}{|\vec{a} \times \vec{b}| |\vec{b} \times \vec{c}|}$

The sign of the dihedral is then found from the triple scalar product calculated by evaluating the determinant of the matrix:

$$\begin{vmatrix} a(x) & a(y) & a(z) \\ b(x) & b(y) & b(z) \\ c(x) & c(y) & c(z) \end{vmatrix} \rightarrow \begin{vmatrix} a(x) & a(y) & a(z) \\ b(x) & b(y) & b(z) \\ c(x) & c(y) & c(z) \end{vmatrix}$$

\*

which is:  $a[x] * (b[y]*c[z]-c[y]*b[z]) - a[y]*(b[x]*c[z]-c[x]*b[z]) + a[z]*(b[x]*c[y]-c[x]*b[y])$

#### Returns

The angle between the input atoms, in radians

#### 4.18.2.4 void calc\_fit\_dimensions ( global\_options\_struct \* global\_options, parm\_struct \* parm\_dats )

Calculates the total number of parameters to be fit- the dimensionality of the problem. TODO - update with multiple prmtops as right now it is a simple loop

##### Parameters

|         |                       |                                                                  |
|---------|-----------------------|------------------------------------------------------------------|
| in, out | <i>global_options</i> | The global options structure, where NDIMENSIONS is to be updated |
| in      | <i>parm_dats</i>      | Array of parameter data structures                               |

#### 4.18.2.5 int calculate\_no\_fit\_params ( parm\_struct \* parm\_data, short int MODE )

Calculates the number of parameters to be fit according to MODE Used to find number of bond, angle, and dihedral params separately.

##### Parameters

|    |                  |                                                                |
|----|------------------|----------------------------------------------------------------|
| in | <i>parm_data</i> | The parameter structure to examine                             |
| in | <i>MODE</i>      | BONDS, ANGLES, or DIHEDRALS indicating parameters to calculate |

#### Returns

The number of parameters of type MODE that are to be fit

#### 4.18.2.6 int check\_for\_valid\_filename ( const char \* data\_string, const int length )

Checks for invalid characters in a filename string.

##### Parameters

|    |                    |                          |
|----|--------------------|--------------------------|
| in | <i>data_string</i> | The string to check      |
| in | <i>length</i>      | The length of the string |

#### Returns

Integer representing SUCCESS or INVALID\_FORMAT

#### 4.18.2.7 int compare\_energy ( const void \* a, const void \* b )

Compare function for qsort in the coordinate structures. Used to compare two coords structures by energy. Please don't call this directly, just pass a function pointer to qsort.

#### 4.18.2.8 int dihedral\_types\_equal ( dihedral\_type\_struct \* first, dihedral\_type\_struct \* second )

Checks if two dihedral types are the same or different. Checks with name only, not by the value of the parameters, and only checks in one direction.

## Parameters

|    |               |                                                                  |
|----|---------------|------------------------------------------------------------------|
| in | <i>first</i>  | Pointer to first <a href="#">dihedral_type_struct</a> to examine |
| in | <i>second</i> | Pointer to <a href="#">dihedral_type_struct</a> to compare to    |

## Returns

YES if they are the same, NO if not

4.18.2.9 int find\_flag ( FILE \* *fptr*, char \* *label* )

Used to locate a title in a prmtop. Leaves the file pointer positioned at the line below the selected label. Rewinds the file pointer before searching.

## Parameters

|         |              |                         |
|---------|--------------|-------------------------|
| in, out | <i>fptr</i>  | The file pointer to use |
| in      | <i>label</i> | The title to find       |

## Returns

Integer SUCCESS, or INVALID\_LINE if label not found.

4.18.2.10 void handle\_sigint ( int *param* )

Prints an error message on a signal. This may someday print something more useful.

## Parameters

|    |               |                           |
|----|---------------|---------------------------|
| in | <i>signal</i> | The signal that is caught |
|----|---------------|---------------------------|

4.18.2.11 int modify\_params\_scratch\_data ( global\_options\_struct \* *global\_options*, parm\_struct \* *parm\_data*, double \* *parameters*, readwrite\_t *MODE* )

Reads and writes parameters in the [parm\\_struct](#) data structure. Extracts parameters marked as variable from the prmtop and puts them in the linear array, or takes parameters from the array and puts them in the data structure. The order in which the extraction is done is as follows: K BOND x (KR, KEQ) BOND y (KR, KEQ) ANGLE x (KT, THEQ) ANGLE y (KT, THEQ) DIHEDRAL x (KP, PN, PHASE) DIHEDRAL y (KP, PN, PHASE)

## Parameters

|         |                       |                                                                                                  |
|---------|-----------------------|--------------------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                                                     |
| in, out | <i>parm_data</i>      | The parameters data structure to read or write to                                                |
| in, out | <i>parameters</i>     | Pre-allocated array size NDIMENSIONS to read or write to                                         |
| in      | <i>MODE</i>           | if READ, copy the data from parm_data to parameters. if WRITE, copy from parameter to parm_data. |

## Returns

The number of parameters successfully extracted.

I am aware that this whole procedure here is clunky and slow but it makes it significantly easier to understand and debug. At some point I will replace this whole system with a much more efficient method

#### 4.18.2.12 int name\_copy ( FILE \* *fptr*, char \* *stringp* )

Desigend to read 4 characters from a file stream and null terminate the string.

#### See Also

[prmtop\\_params.h](#) for definition of `NAME_SIZE`

#### Parameters

|                |                |                           |
|----------------|----------------|---------------------------|
| <i>in, out</i> | <i>fptr</i>    | File pointer to read from |
| <i>in, out</i> | <i>stringp</i> | The string to read into   |

#### Returns

Integer indicating SUCCESS or INVALID\_LINE

#### 4.18.2.13 int not\_enough\_dihedrals ( parm\_struct \* *parm\_data*, int *n* )

Potentially adds more dihedral terms to existing ones so you can get a better fit. This is completely deprecated by dihedral data refactoring and so is commented out for now.

#### 4.18.2.14 int ObfuscateAtom ( int *at* )

Re-obfuscates an atom from atomic number into prmtop format. This can be used in writing a prmtop.

#### See Also

[unObfuscateAtom](#)

#### Parameters

|           |           |                       |
|-----------|-----------|-----------------------|
| <i>in</i> | <i>at</i> | The atom to obfuscate |
|-----------|-----------|-----------------------|

#### Returns

The prmtop obfuscated atom number.

#### 4.18.2.15 void print\_backtrace ( int *signal* )

Prints a backtrace for debugging. This does not work in Cygwin and so is ifdef'd out in that case.

#### Parameters

|           |               |                           |
|-----------|---------------|---------------------------|
| <i>in</i> | <i>signal</i> | The signal that is caught |
|-----------|---------------|---------------------------|

#### 4.18.2.16 void print\_close\_line\_box ( int *no\_spaces* )

Prints the closing of a line in a box styled like

```
||
```

See Also

[print\\_open\\_line\\_box](#)

#### Parameters

|           |                  |                                            |
|-----------|------------------|--------------------------------------------|
| <i>in</i> | <i>no_spaces</i> | The number of spaces to go before the line |
|-----------|------------------|--------------------------------------------|

#### 4.18.2.17 void print\_dihedral ( dihedral\_type\_struct \* type, int term )

Pretty prints one dihedral value and parameters

#### Parameters

|           |             |                                                                  |
|-----------|-------------|------------------------------------------------------------------|
| <i>in</i> | <i>type</i> | Pointer to the <a href="#">dihedral_type_struct</a> to print     |
| <i>in</i> | <i>term</i> | Which term of this dihedral the parameters should be printed for |

#### 4.18.2.18 void print\_open\_line\_box ( int \* i )

Prints the beginning of an ASCII box.

See Also

[print\\_close\\_line\\_box](#)

#### Parameters

|                |          |                                       |
|----------------|----------|---------------------------------------|
| <i>in, out</i> | <i>i</i> | Contains number of characters printed |
|----------------|----------|---------------------------------------|

#### 4.18.2.19 int s\_getline ( char \* line, int max, FILE \* fp )

"Safe" version of getline, exits with an error if EOF is found.

#### Parameters

|                |             |                                              |
|----------------|-------------|----------------------------------------------|
| <i>in, out</i> | <i>line</i> | String that will contain the read line       |
| <i>in</i>      | <i>max</i>  | The maximum number of characters to read     |
| <i>in</i>      | <i>fp</i>   | File pointer to the file to read a line from |

#### Returns

The number of characters read, or FILE\_READ\_FAIL if EOF found

#### 4.18.2.20 int unObfuscateAtom ( int at )

Translates atom numbers from prmtop format into understandable numbers. PARM for some reason obfuscates each atom number by (when positive),  $at = (at+3) / 3$ . This function returns the true atom number. For example,  $unObfuscateAtom(24) = 9$ .

See Also

[ObfuscateAtom](#)

## Parameters

|    |    |                                  |
|----|----|----------------------------------|
| in | at | The obfuscated atom to translate |
|----|----|----------------------------------|

## Returns

The true atom number

**4.18.2.21** `int update_prmtop_data ( global_options_struct * global_options, parm_struct * parm_datas, double * parameters )`

Puts the parameters in a data array into each prmtop. Used in multi-prmtop fits to update each of them before conducting an energy evaluation.

## Parameters

|         |                       |                                                          |
|---------|-----------------------|----------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                             |
| in, out | <i>parm_data</i>      | Array of parameter data structures that will be updated. |
| in      | <i>parameters</i>     | Pre-allocated array size NDIMENSIONS to read from        |

## Returns

Integer indicating success or failure.

## 4.19 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/options\_summary.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "function_def.h"
```

## Functions

- void `print_job_control_summary (global_options_struct *global_options, parm_struct *parm_datas, coord_set *coords_datas)`

### 4.19.1 Detailed Description

Prints out routines for printing options summaries at program start

### 4.19.2 Function Documentation

**4.19.2.1** `void print_job_control_summary ( global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_datas )`

Prints a summary of the options in the job control file

## Parameters

|    |                       |                                              |
|----|-----------------------|----------------------------------------------|
| in | <i>global_options</i> | The global options structure                 |
| in | <i>parm_data</i>      | Array of parameter files                     |
| in | <i>coords_datas</i>   | Array of coordinate sets of input structures |



## 4.20 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/param\_summary.c File Reference

```
#include <stdio.h>
#include "function_def.h"
#include "constants.h"
```

### Functions

- void [print\\_multiprmtop\\_summary](#) ([global\\_options\\_struct](#) \*global\_options, [parm\\_struct](#) \*parm\_datas)
- void [print\\_parameter\\_summary](#) ([global\\_options\\_struct](#) \*global\_options, [parm\\_struct](#) \*parm\_data)

#### 4.20.1 Detailed Description

Prints out a summary of all the parameters

#### 4.20.2 Function Documentation

4.20.2.1 void [print\\_multiprmtop\\_summary](#) ( [global\\_options\\_struct](#) \* *global\_options*, [parm\\_struct](#) \* *parm\_datas* )

Prints out a summary of only the parameters to be fit over a set of multiple prmtops. Assumes that each prmtop contains every single parameter to be fit, because only the data from the first one is used.

See Also

[verify\\_prmtops](#)

#### Parameters

|    |                       |                               |
|----|-----------------------|-------------------------------|
| in | <i>global_options</i> | The global options structure  |
| in | <i>parm_datas</i>     | Array of parameter structures |

4.20.2.2 void [print\\_parameter\\_summary](#) ( [global\\_options\\_struct](#) \* *global\_options*, [parm\\_struct](#) \* *parm\_data* )

Prints out a summary of all parameters currently stored in one *parm\_data*. A \* will mark which parameters are being fit.

#### Parameters

|                       |                                  |
|-----------------------|----------------------------------|
| <i>global_options</i> | The global options structure     |
| <i>parm_data</i>      | The parameter structure to print |

## 4.21 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/parameter\_optimiser.c File Reference

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <signal.h>
#include "function_def.h"
```

## Functions

- int [main](#) (int argc, char \*argv[])

### 4.21.1 Function Documentation

4.21.1.1 int main ( int argc, char \* argv[] )

## 4.22 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/print\_program\_info.c File Reference

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

## Functions

- void [print\\_program\\_info](#) (void)
- void [print\\_program\\_history](#) (void)

### 4.22.1 Function Documentation

4.22.1.1 void print\_program\_history ( void )

4.22.1.2 void print\_program\_info ( void )

## 4.23 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/prmtop\_params.h File Reference

## Data Structures

- struct [atom\\_struct](#)
- struct [residue](#)
- struct [parmbond\\_struct](#)
- struct [parmangle\\_struct](#)
- struct [parmdihedral\\_struct](#)
- struct [bond\\_data\\_struct](#)
- struct [angle\\_data\\_struct](#)
- struct [dihedral\\_data\\_struct](#)
- struct [dihedral\\_type\\_struct](#)
- struct [parm\\_struct](#)

## Macros

- #define [BUFFER\\_SIZE](#) 1024
- #define [PRMTOP\\_TITLE\\_LENGTH](#) 81
- #define [MAX\\_BONDS\\_PER\\_TYPE](#) 256
- #define [MAX\\_ANGLES\\_PER\\_TYPE](#) 512

- `#define MAX_DIHEDRALS_PER_TYPE 1024`
- `#define NAME_SIZE 5`
- `#define NAME_DEFAULT " "`

## Typedefs

- `typedef char Name [NAME_SIZE]`

### 4.23.1 Detailed Description

Contains parameters pertaining to prmtop files, including information necessary to reconstruct them, and the data structures into which the prmtop bonds, angles, and dihedrals are inserted in order to make them ready for fitting

Information is duplicated in two forms— the parm for (to allow re-creation of the parm file if required at some point), and the more obvious form, where each "bond" has associated parameters and optional scale factors.

### 4.23.2 Macro Definition Documentation

#### 4.23.2.1 `#define BUFFER_SIZE 1024`

Length of buffer for reading in prmtop data

#### 4.23.2.2 `#define MAX_ANGLES_PER_TYPE 512`

Maximum number of angles of a given type allowed

#### 4.23.2.3 `#define MAX_BONDS_PER_TYPE 256`

Maximum number of bonds of a given type allowed

#### 4.23.2.4 `#define MAX_DIHEDRALS_PER_TYPE 1024`

Maximum number of dihedrals of a given type allowed

#### 4.23.2.5 `#define NAME_DEFAULT " "`

#### 4.23.2.6 `#define NAME_SIZE 5`

The parameter file assumes the atom, residue, symbol, etc. names to be \* four characters (we will store them as strings, requiring a null terminator, \* hence the size is 5).

#### 4.23.2.7 `#define PRMTOP_TITLE_LENGTH 81`

Title length of old format prmtop file

### 4.23.3 Typedef Documentation

#### 4.23.3.1 `typedef char Name [NAME_SIZE]`

## 4.24 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/process\_command\_line.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "function_def.h"
```

### Macros

- `#define MAX_CMDLINE_OPTIONS 13`

### Functions

- `int process_command_line (int argc, char **argv, global_options_struct *global_options, parm_struct **parm_datas, coord_set **coords_datas)`

#### 4.24.1 Detailed Description

Utilities for processing command line options

#### 4.24.2 Macro Definition Documentation

##### 4.24.2.1 `#define MAX_CMDLINE_OPTIONS 13`

Prog name + 12 options

#### 4.24.3 Function Documentation

##### 4.24.3.1 `int process_command_line ( int argc, char ** argv, global_options_struct * global_options, parm_struct ** parm_datas, coord_set ** coords_datas )`

Processes all command line options. Possible options combinations are: paramfit -i [Job control file] -p [prmtop] -c [mdcrd] -q [quantum data] -d [ON/OFF/DEBUG] --random-seed [seed] paramfit -i [Job control file] -pf [prmtop list] -cf [mdcrd list] -qf [quantum list] -d [ON/OFF/DEBUG] --random-seed [seed]

#### Parameters

|         |                       |                                                                                           |
|---------|-----------------------|-------------------------------------------------------------------------------------------|
| in      | <i>argv</i>           | Pointer to array with all options, argv[0] is the name of the program (paramfit)          |
| in, out | <i>global_options</i> | Global options structure that will be updated                                             |
| in, out | <i>parm_datas</i>     | Unallocated array of parm structs, will be updated with initial blank one if -p specified |
| in, out | <i>coords_datas</i>   | Unallocated of coords structs, will be updated with initial blank one if -c specified     |

## Returns

Integer indicating success, failure, options problems, etc.

## 4.25 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/process\_job\_control\_setting.c File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "function_def.h"

```

## Functions

- int [process\\_job\\_control\\_setting](#) (char \*setting\_line, int length, int \*number\_settings, [global\\_options\\_struct](#) \*global\_options, [coord\\_set](#) \*coords\_data)

### 4.25.1 Detailed Description

Includes big function to process one line of a job control file.

### 4.25.2 Function Documentation

4.25.2.1 int [process\\_job\\_control\\_setting](#) ( char \* *setting\_line*, int *length*, int \* *number\_settings*, [global\\_options\\_struct](#) \* *global\_options*, [coord\\_set](#) \* *coords\_data* )

Processes one line of the job control file and sets corresponding options. This function is called repeatedly for each line of the job control file until all settings are processed.

## See Also

[read\\_job\\_control\\_file](#)

## Parameters

|         |                        |                                                                                                                                                                                      |
|---------|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in      | <i>setting_line</i>    | The raw line from the job control file as a character array                                                                                                                          |
| in      | <i>length</i>          | The length of this line                                                                                                                                                              |
| in, out | <i>number_settings</i> | The number of settings that have been set. Will be updated if successful.                                                                                                            |
| in, out | <i>global_options</i>  | The global options structure to input settings to                                                                                                                                    |
| in, out | <i>coords_data</i>     | Pointer to array of coords struct already allocated to size 1 that may have its number of structures set if NSTRUCTURES is specified in the job control file, for single prmtop fits |

## Returns

Integer indicating success or various types of failure (invalid setting/data, alloc fail, etc)

## 4.26 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/process\_prmtop.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "function_def.h"
#include "constants.h"
```

## Functions

- int [process\\_prmtops](#) ([global\\_options\\_struct](#) \*global\_options, [parm\\_struct](#) \*parm\_datas)
- int [verify\\_prmtops](#) ([global\\_options\\_struct](#) \*global\_options, [parm\\_struct](#) \*parm\_datas)
- int [bondcomparator](#) (const void \*a, const void \*b)
- int [anglecomparator](#) (const void \*a, const void \*b)
- int [dihedralcomparator](#) (const void \*a, const void \*b)
- int [process\\_single\\_prmtop](#) ([global\\_options\\_struct](#) \*global\_options, [parm\\_struct](#) \*parm\_data)
- void [set\\_dihedral\\_fit](#) ([global\\_options\\_struct](#) \*global\_options, [dihedral\\_type\\_struct](#) \*s, int t, [bool\\_t](#) d\_kp, [bool\\_t](#) d\_np, [bool\\_t](#) d\_phase)

### 4.26.1 Detailed Description

This routine is responsible for processing the prmtop file into separate parameters for each bond, angle and dihedral type.

This is necessary because the prmtop file and thus prmtop storage arrays do not keep separate parameters for different bond / angle or dihedral setups that share the same values for these parameters

e.g. O-C-N-H and O-C-N-CH3 share the same parameters and so this is stored in the same spot in memory We need to split this in order to optimise them individually

## See Also

[read\\_prmtop.c](#)

### 4.26.2 Function Documentation

#### 4.26.2.1 int anglecomparator ( const void \* a, const void \* b )

Compares names of two angles, for qsort

#### 4.26.2.2 int bondcomparator ( const void \* a, const void \* b )

Compares names of two bonds, for qsort.

#### 4.26.2.3 int dihedralcomparator ( const void \* a, const void \* b )

Compares names of two dihedrals, for qsort

4.26.2.4 `int process_prmtops ( global_options_struct * global_options, parm_struct * parm_dats )`

Processes all the prmtop data. Processes each individual [parm\\_struct](#).

## Parameters

|                      |                             |                                                           |
|----------------------|-----------------------------|-----------------------------------------------------------|
| <code>in</code>      | <code>global_options</code> | The global options structure, specifies number of prmtops |
| <code>in, out</code> | <code>parm_dats</code>      | Array of parm structures to process                       |

## Returns

Integer indicating success or failure

4.26.2.5 `int process_single_prmtop ( global_options_struct * global_options, parm_struct * parm_data )`

Processes raw prmtop data into arrays of unique bond, angle, and dihedral parameters that are in structures that are much easier to optimize. Done on one prmtop at a time

## See Also

[read\\_prmtops](#)  
[process\\_prmtops](#)

## Parameters

|                      |                             |                                                                                                  |
|----------------------|-----------------------------|--------------------------------------------------------------------------------------------------|
| <code>in, out</code> | <code>global_options</code> | The global options structure. <code>unique_[bonds,angles,dihedrals]_found</code> will be updated |
| <code>in, out</code> | <code>parm_data</code>      | Contains the raw prmtop data, will also be populated with processed prmtop data                  |

## Returns

Integer indicating success or failure

4.26.2.6 `void set_dihedral_fit ( global_options_struct * global_options, dihedral_type_struct * s, int t, bool_t d_kp, bool_t d_np, bool_t d_phase )`

Sets dihedral fitting options for a given dihedral. All parameters for this term should have already been initialized.

## Parameters

|                      |                             |                                                     |
|----------------------|-----------------------------|-----------------------------------------------------|
| <code>in</code>      | <code>global_options</code> | The global options structure with prompting options |
| <code>in, out</code> | <code>s</code>              | The dihedral type to set options for                |
| <code>in</code>      | <code>t</code>              | Integer indicating the term to fit                  |
| <code>in</code>      | <code>d_kp</code>           | Whether to prompt to fit dihedral KP                |
| <code>in</code>      | <code>d_np</code>           | Whether to prompt to fit dihedral NP                |
| <code>in</code>      | <code>d_phase</code>        | Whether to prompt to fit dihedral PHASE             |

4.26.2.7 `int verify_prmtops ( global_options_struct * global_options, parm_struct * parm_dats )`

Checks that each prmtop contains all of the parameters that are to be fit. This is necessary for consistency in the fitting, as there is no point in doing a multi-molecule fit if a parameter is only present in one of the molecules.

## Parameters

|    |                       |                              |
|----|-----------------------|------------------------------|
| in | <i>global_options</i> | The global options structure |
| in | <i>parm_datas</i>     | Array of parameter structure |

## Returns

Integer indicating success (prmtops okay) or failure (inconsistency detected or other error)

## 4.27 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/read\_energy.c File Reference

```
#include <stdlib.h>
#include <sys/stat.h>
#include <string.h>
#include "constants.h"
#include "function_def.h"
```

## Functions

- `int read_qm (global_options_struct *global_options, parm_struct *parm_datas, coord_set *coords_datas)`
- `int read_qm_energy_list (global_options_struct *global_options, parm_struct *parm_data, coord_set *coords_data)`
- `int read_qm_directory (global_options_struct *global_options, parm_struct *parm_data, coord_set *coords_data)`
- `int read_gaussian_forces (global_options_struct *global_options, parm_struct *parm_data, coord_set *coords_data)`
- `int read_gaussian_energy (global_options_struct *global_options, parm_struct *parm_data, coord_set *coords_data)`

### 4.27.1 Detailed Description

Contains routines for reading in energy data into the coordinate structures. This includes support for reading in each format of QM file, AMBER files (TODO) and straight up lists of energies. This is all handled by the unified function `read_qm` which calls the appropriate function to the qm data type.

### 4.27.2 Function Documentation

**4.27.2.1** `int read_gaussian_energy ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data )`

Reads in the gaussian energy from a certain set of files into a coordinate structure This lets paramfit write the file, run gaussian, read the file without any need for the user to go through the gaussian output.

## Parameters

|         |                       |                                                    |
|---------|-----------------------|----------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                       |
| in      | <i>parm_data</i>      | Pointer to single parm struct                      |
| in, out | <i>coords_data</i>    | Pointer to single coord struct to attach energy to |

## Returns

Integer indicating success or failure



4.27.2.2 `int read_gaussian_forces ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data )`

Reads in Gaussian output files with forces for all structures. The files need to have been generated in the CREATE\_INPUT mode so that the atom numbering is consistent. This also lets us assume that the files are named QMFILEOUTSTART###QMFILEOUTEND.out

#### Parameters

|         |                       |                                                                    |
|---------|-----------------------|--------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                       |
| in      | <i>parm_data</i>      | Pointer to single parameter file                                   |
| in, out | <i>coords_data</i>    | Pointer to single coordinate set, will have forces attached to it. |

#### Returns

Integer indicating success or failure.

4.27.2.3 `int read_qm ( global_options_struct * global_options, parm_struct * parm_datas, coord_set * coords_datas )`

The master function for all reading of QM input data. Will call functions to read in either forces or energies from a list or from raw output files from the appropriate program.

#### Parameters

|         |                       |                                                                                                                                            |
|---------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure, which specifies the data format and energy or forces                                                         |
| in      | <i>parm_datas</i>     | Array of parameter sets corresponding to coordinates to read in                                                                            |
| in, out | <i>coords_datas</i>   | Array of coordinate sets that each have the filename/folder to read from initialized. These will be updated with the QM energies or forces |

#### Returns

Integer indicating success or failure

4.27.2.4 `int read_qm_directory ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data )`

Reads all qm output files in a given directory. The names of the files are according to the convention paramfit writes them in with CREATE\_INPUT mode- that is prmtop.QMFILEOUTEND.## where prmtop is prmtop->filename. This will support a variety of file formats, which one it is is set in global\_options->QMFORMAT or similar.

#### Parameters

|         |                       |                                                                                                                                          |
|---------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                                                                                             |
| in, out | <i>coords_data</i>    | Pointer to single coordinate set. Each structure in the set will be populated with the energy / forces from the matching QM output file. |

**Returns**

Integer indicating success or failure

**4.27.2.5** `int read_qm_energy_list ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data )`

Reads in the quantum energies and attaches one to each coordinate structure. Performs unit conversion if necessary. This is used when evaluating energies, NOT forces. The QM data file should consist of NSTRUCTURES worth of doubles, with one value per line.

**Parameters**

|         |                       |                                                                            |
|---------|-----------------------|----------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure, containing filename with energies            |
| in      | <i>parm_data</i>      | Pointer to single parameter structure corresponding to this coordinate set |
| in, out | <i>coords_data</i>    | Pointer to single coordinate set that energies will be attached to.        |

**Returns**

Integer indicating success or failure.

## 4.28 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/read\_mdcrd.c File Reference

```
#include "function_def.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

**Functions**

- `int read_mdcrds (global_options_struct *global_options, parm_struct *parm_datas, coord_set **s_datas)`
- `int read_single_mdcrd (coord_set *coords_data)`

### 4.28.1 Detailed Description

Contains functions relating to reading and writing coordinate structures.

### 4.28.2 Function Documentation

**4.28.2.1** `int read_mdcrds ( global_options_struct * global_options, parm_struct * parm_datas, coord_set ** s_datas )`

Reads in each coordinate file from the list of coordinate file.

**Parameters**

|     |                       |                                                                                                                                                                          |
|-----|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in  | <i>global_options</i> | The global options structure                                                                                                                                             |
| in  | <i>parm_datas</i>     | The array of parameter sets to use                                                                                                                                       |
| out | <i>struc_datas</i>    | Pointer to array of structure sets to be initialized. The array is size 1 already if -c command line option for single mdcrd has been specified, otherwise *s_datas=NULL |

**Returns**

Integer indicating success or failure

**4.28.2.2 int read\_single\_mdcrd ( coord\_set \* coords\_data )**

Reads the coordinate file with all the input structures.

**Parameters**

|                      |                          |                                                                                                                                                                                         |
|----------------------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>in, out</code> | <code>coords_data</code> | Pointer to coordinate set structure that will be populated. Filename and number of structures needs to be already initialized. The coordinate data set should also have been allocated. |
|----------------------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**See Also**

[alloc\\_coords](#)  
[read\\_mdcrds](#)

**Returns**

Integer indicating success or failure.

**4.29 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/read\_prmtop.c File Reference**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "function_def.h"
```

**Functions**

- int [read\\_prmtops](#) (global\_options\_struct \*global\_options, parm\_struct \*\*parm\_datas)
- int [read\\_single\\_prmtop](#) (global\_options\_struct \*global\_options, parm\_struct \*parm\_data)

**4.29.1 Detailed Description**

The routines here are responsible for opening, reading and processing the prmtop file.

Note, we allocate memory and read all of the options in the prmtop file even though a number, such as hydrogen bonding are not actually used in the fitting. The reason for this is that it means other parts of the program can be updated at a later date without worrying if the data is actually read from the prmtop file.

**4.29.2 Function Documentation****4.29.2.1 int read\_prmtops ( global\_options\_struct \* global\_options, parm\_struct \*\* parm\_datas )**

Reads in all prmtop data. Starts by getting filenames one line at a time from the prmtop list and then reading in data from each one of those.

**See Also**

[read\\_single\\_prmtop](#)

**Parameters**

|         |                       |                                                           |
|---------|-----------------------|-----------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure, with the parm list filename |
| in, out | <i>parm_data</i>      | Pointer to an array of parm_structs, will be reallocd.    |

**Returns**

Integer indicating success or failure

#### 4.29.2.2 int read\_single\_prmtp ( global\_options\_struct \* *global\_options*, parm\_struct \* *parm\_data* )

Reads the raw data from the prmtp file. Sorts out how many atoms, atom types, and basic info. Does not do any processing into the fitting data structure.

**See Also**

[process\\_prmtp.c](#)

**Parameters**

|         |                                         |                                                    |
|---------|-----------------------------------------|----------------------------------------------------|
| in      | <i>global_options</i>                   | The global options structure                       |
| in, out | <i>parm_data</i>                        | The parameter structure that will have info put in |
| in      | <i>parm_data</i> -<br>> <i>filename</i> | The path to the prmtp file to read                 |

## 4.30 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/simplex.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "function_def.h"
```

**Functions**

- int [minimise\\_function\\_simplex](#) (global\_options\_struct \*global\_options, parm\_struct \*parm\_data, coord\_set \*coords\_data)

### 4.30.1 Detailed Description

Contains the simplex minimization algorithm Requires (N+1)\*N (doubles) Storage for the simplex array as well as (3\*NDIMENSIONS)+10 doubles as scratch.

### 4.30.2 Function Documentation

#### 4.30.2.1 int minimise\_function\_simplex ( global\_options\_struct \* *global\_options*, parm\_struct \* *parm\_data*, coord\_set \* *coords\_data* )

The simplex minimization function. Written mostly by Ross Walker

## Parameters

|         |                       |                                                                           |
|---------|-----------------------|---------------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                              |
| in, out | <i>parm_data</i>      | Array of parameter structures, updated at each iteration                  |
| in      | <i>coords_data</i>    | Array of coordinate sets containing all the input structures with QM data |

## Returns

Integer indicating success or failure

## 4.31 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/wizard.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include "function_def.h"
```

## Functions

- int [job\\_control\\_wizard](#) ([global\\_options\\_struct](#) \*global\_options)
- int [get\\_option](#) (int min, int max)
- double [get\\_float](#) ()
- void [genetic\\_wizard](#) ([global\\_options\\_struct](#) \*global\_options, FILE \*file)
- void [simplex\\_wizard](#) ([global\\_options\\_struct](#) \*global\_options, FILE \*file)

## 4.31.1 Function Documentation

4.31.1.1 void [genetic\\_wizard](#) ( [global\\_options\\_struct](#) \* *global\_options*, FILE \* *file* )

Reads in options specific to the genetic algorithm

## Parameters

|         |                       |                                                |
|---------|-----------------------|------------------------------------------------|
| in, out | <i>global_options</i> | Structure where options will be set            |
| in, out | <i>file</i>           | File to save options to, or NULL if not saving |

4.31.1.2 double [get\\_float](#) ( )

Gets a floating point value from stdin

## Returns

The value that was read, as a double

4.31.1.3 int [get\\_option](#) ( int *min*, int *max* )

Gets an integer in the specified range from stdin

## Parameters

|    |            |                                      |
|----|------------|--------------------------------------|
| in | <i>min</i> | The minimum allowed value, exclusive |
| in | <i>max</i> | The maximum allowed value, exclusive |

**Returns**

The integer that was read

**4.31.1.4 int job\_control\_wizard ( global\_options\_struct \* global\_options )**

The main job control wizard. Walks you through the various options and lets you save them as you go.

**Parameters**

|                |                       |                                                      |
|----------------|-----------------------|------------------------------------------------------|
| <i>in, out</i> | <i>global_options</i> | Will contain options set by wizard, file to write to |
|----------------|-----------------------|------------------------------------------------------|

**Returns**

Integer indicating success or failure.

**4.31.1.5 void simplex\_wizard ( global\_options\_struct \* global\_options, FILE \* file )**

Reads in options specific to the simplex algorithm

**Parameters**

|                |                       |                                                |
|----------------|-----------------------|------------------------------------------------|
| <i>in, out</i> | <i>global_options</i> | Structure where options will be set            |
| <i>in, out</i> | <i>file</i>           | File to save options to, or NULL if not saving |

**4.32 /home/rbetz/git\_tree/amber/AmberTools/src/paramfit/write\_input.c File Reference**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "function_def.h"
#include "constants.h"
```

**Functions**

- int [write\\_input\\_gaussian](#) (global\_options\_struct \*global\_options, [parm\\_struct](#) \*parm\_data, [coord\\_set](#) \*current\_struct, int num, FILE \*fptr)
- int [write\\_input\\_parameters](#) (global\_options\_struct \*global\_options, [parm\\_struct](#) \*parm\_data)
- int [write\\_frmod](#) (global\_options\_struct \*global\_options, [parm\\_struct](#) \*parm\_data)
- int [write\\_input\\_adf](#) (global\_options\_struct \*global\_options, [parm\\_struct](#) \*parm\_data, [coord\\_set](#) \*current\_struct, int num, FILE \*fptr)
- int [write\\_input\\_gamess](#) (global\_options\_struct \*global\_options, [parm\\_struct](#) \*parm\_data, [coord\\_set](#) \*current\_struct, int num, FILE \*fptr)
- int [write\\_energy](#) (global\_options\_struct \*global\_options, [parm\\_struct](#) \*parm\_data, [coord\\_set](#) \*coords\_data, int generation)
- int [write\\_prmtp](#) (global\_options\_struct \*global\_options, [parm\\_struct](#) \*parm\_data)

**4.32.1 Detailed Description**

Contains routines for writing output files. Contains routines for writing all quantum input files as well as various other outputs from fitting

## 4.32.2 Function Documentation

**4.32.2.1** `int write_energy ( global_options_struct * global_options, parm_struct * parm_data, coord_set * coords_data, int generation )`

Writes a data file with amber and quantum energies for each structure with given parameters. Useful to plot the results of a calculation to see the quality of fit.

### Parameters

|    |                       |                                                                        |
|----|-----------------------|------------------------------------------------------------------------|
| in | <i>global_options</i> | The global options structure, with filename to save to                 |
| in | <i>parm_data</i>      | Pointer to single parameter set to use when calculating amber energies |
| in | <i>coords_data</i>    | Pointer to single coordinate set with attached quantum energy          |
| in | <i>generation</i>     | The generation of the genetic algorithm, so can be called iteratively  |

### Returns

Integer indicating success or failure

**4.32.2.2** `int write_frcmod ( global_options_struct * global_options, parm_struct * parm_data )`

Write a force field modification file. To be used to save the results of a fit for reading into Leap

### Parameters

|    |                       |                                                            |
|----|-----------------------|------------------------------------------------------------|
| in | <i>global_options</i> | The global options structure, containing filename to write |
| in | <i>parm_data</i>      | The parameter file containing fitted results               |

### Returns

Integer indicating success or failure

**4.32.2.3** `int write_input_adf ( global_options_struct * global_options, parm_struct * parm_data, coord_set * current_struct, int num, FILE * fptr )`

Writes an input file for ADF for a structure. The file is initialized in write quantum. Note that this file is only for energy calculations, not forces.

### Parameters

|         |                       |                                                                     |
|---------|-----------------------|---------------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                        |
| in      | <i>parm_data</i>      | Pointer to the parameter structure that describes these coordinates |
| in      | <i>current_struct</i> | Pointer to the coordinate set that will be used                     |
| in      | <i>num</i>            | Number of the structure in teh coordinate set to write              |
| in, out | <i>fptr</i>           | Pointer to the file to write to, initialized elsewhere              |

### Returns

Integer indicating success or failure

**4.32.2.4** `int write_input_gamess ( global_options_struct * global_options, parm_struct * parm_data, coord_set * current_struct, int num, FILE * fptr )`

Writes a quantum input file for GAMESS. This is for energy calculation only.

**Parameters**

|         |                          |                                                                        |
|---------|--------------------------|------------------------------------------------------------------------|
| in      | <i>global_options</i>    | The global options structure                                           |
| in      | <i>parm_data</i>         | The parameter structure                                                |
| in      | <i>current_structure</i> | Pointer to coordinate set to write from                                |
| in      | <i>num</i>               | The number of the structure in the coordinate set that will be written |
| in, out | <i>fptr</i>              | File to write to, should be already initialized                        |

**Returns**

Integer indicating success or failure.

**4.32.2.5** `int write_input_gaussian ( global_options_struct * global_options, parm_struct * parm_data, coord_set * current_struct, int num, FILE * fptr )`

Writes a gaussian input file for one structure. Should not be called directly. Uses a header file containing all of the quantum options that the atom coordinates are appended to, so this can be used to make input files for force or energy calculations.

**See Also**

`write_input`

**Parameters**

|         |                       |                                                              |
|---------|-----------------------|--------------------------------------------------------------|
| in      | <i>global_options</i> | The global options structure                                 |
| in      | <i>parm_data</i>      | The parameter file                                           |
| in      | <i>current_struct</i> | Pointer to coordinate set to use                             |
| in      | <i>num</i>            | The number of the structure to write from the coordinate set |
| in, out | <i>fptr</i>           | Pointer to the file to write to                              |

**Returns**

Integer indicating success or failure

**4.32.2.6** `int write_input_parameters ( global_options_struct * global_options, parm_struct * parm_data )`

Writes a file of which parameters are to be fit. This can be read in later to fit multiple runs with one set of parameters to fit. Assumes if multiple prmtops, the parameters are the same in each one

**Parameters**

|    |                       |                                                              |
|----|-----------------------|--------------------------------------------------------------|
| in | <i>global_options</i> | Global options structure, containing filename to save ase    |
| in | <i>parm_data</i>      | The parameter file containing which parameters are to be fit |

**Returns**

Integer indicating success or failure

**4.32.2.7** `int write_prmtp ( global_options_struct * global_options, parm_struct * parm_data )`

Saves a prmtp with the given parameters. Really experimental, and I'm not sure if this is even useful.



**Parameters**

|    |                       |                                                             |
|----|-----------------------|-------------------------------------------------------------|
| in | <i>global_options</i> | The global options structure, including the file to save to |
| in | <i>parm_data</i>      | The parameters to put into the prmtop file                  |

**Returns**

Integer indicating success or failure

# Index

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/bound-  
\_check.c, 31

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/calc-  
\_r\_squared.c, 33

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/constan-  
h, 34

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/create-  
\_input.c, 34

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/default-  
c, 35

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/dihedral-  
\_fitting.c, 36

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/element-  
c, 37

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/error-  
\_messages.c, 37

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/eval-  
\_amber\_forces.c, 39

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/eval-  
\_amber\_std.c, 41

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/file-  
\_io.c, 43

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/fitting-  
\_control.c, 44

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/forces-  
h, 45

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/function-  
\_def.h, 45

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/genetic-  
\_algorithm.c, 77

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/help-  
\_functions.c, 79

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/mem-  
\_alloc.c, 79

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/misc-  
\_utils.c, 80

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/options-  
\_summary.c, 86

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/param-  
\_summary.c, 87

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/parameter-  
\_optimiser.c, 87

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/print-  
\_program\_info.c, 88

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/prmtop-  
\_params.h, 88

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/process-  
\_command\_line.c, 90

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/process-  
\_job\_control\_setting.c, 91

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/process-  
\_prmtop.c, 92

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/read-  
\_energy.c, 94

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/read-  
\_mdcrd.c, 96

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/read-  
\_prmtop.c, 97

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/simplex-  
c, 98

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/wizard-  
c, 99

/home/rbetz/git\_tree/amber/AmberTools/src/paramfit/write-  
\_input.c, 100

\_GNU\_SOURCE

\_function\_def.h, 48

ADF

\_function\_def.h, 51

AMBER\_FORCES

\_function\_def.h, 50

ABORT

\_function\_def.h, 48

ALGORITHM

\_global\_options\_struct, 15

ALLOC\_FAIL

\_function\_def.h, 48

AMBER\_SYSTEM\_CHARGE

\_parm\_struct, 21

ANGLE\_LIMIT

\_global\_options\_struct, 15

ANGLE\_PARAMS

\_global\_options\_struct, 15

ANGLEEQ\_dx

\_global\_options\_struct, 15

ANGLEFC\_dx

\_global\_options\_struct, 15

ANGLES

\_function\_def.h, 48

ag

\_parm\_struct, 21

\_algorithm\_t

\_function\_def.h, 50

\_alloc\_2D\_double

\_function\_def.h, 52

\_mem\_alloc.c, 79

\_alloc\_coords

\_function\_def.h, 52

- mem\_alloc.c, 79
- alloc\_data\_matrix
  - function\_def.h, 52
  - genetic\_algorithm.c, 77
- amass
  - atom\_struct, 7
- angle\_data
  - parm\_struct, 21
- angle\_data\_struct, 5
  - atom1, 5
  - atom2, 5
  - atom3, 5
  - atom\_type1, 5
  - atom\_type2, 5
  - atom\_type3, 6
  - DO\_FIT\_KT, 6
  - DO\_FIT\_THEQ, 6
  - number, 6
  - teq, 6
  - tk, 6
- angle\_thetas
  - bounds\_struct, 9
- anglecomparator
  - function\_def.h, 52
  - process\_prmtop.c, 92
- atom
  - parm\_struct, 21
- atom1
  - angle\_data\_struct, 5
  - bond\_data\_struct, 8
  - dihedral\_data\_struct, 12
- atom2
  - angle\_data\_struct, 5
  - bond\_data\_struct, 8
  - dihedral\_data\_struct, 12
- atom3
  - angle\_data\_struct, 5
  - dihedral\_data\_struct, 12
- atom4
  - dihedral\_data\_struct, 12
- atom\_struct, 6
  - amass, 7
  - chrg, 7
  - iac, 7
  - igraph, 7
  - irotat, 7
  - isymb1, 7
  - itree, 7
  - join, 7
  - numex, 7
  - res, 7
- atom\_type1
  - angle\_data\_struct, 5
  - bond\_data\_struct, 8
  - dihedral\_type\_struct, 13
- atom\_type2
  - angle\_data\_struct, 5
  - bond\_data\_struct, 8
- dihedral\_type\_struct, 13
- atom\_type3
  - angle\_data\_struct, 6
  - dihedral\_type\_struct, 13
- atom\_type4
  - dihedral\_type\_struct, 13
- BOTH
  - function\_def.h, 50
- BOHR\_TO\_ANGSTROM
  - constants.h, 34
- BOND\_LIMIT
  - global\_options\_struct, 16
- BOND\_PARAMS
  - global\_options\_struct, 16
- BONDEQ\_dx
  - global\_options\_struct, 16
- BONDFC\_dx
  - global\_options\_struct, 16
- BONDS
  - function\_def.h, 48
- BUFFER\_SIZE
  - prmtop\_params.h, 89
- bg
  - parm\_struct, 21
- bond\_data
  - parm\_struct, 21
- bond\_data\_struct, 7
  - atom1, 8
  - atom2, 8
  - atom\_type1, 8
  - atom\_type2, 8
  - DO\_FIT\_KR, 8
  - DO\_FIT\_REQ, 8
  - number, 8
  - req, 8
  - rk, 8
- bond\_lengths
  - bounds\_struct, 9
- bondcomparator
  - function\_def.h, 52
  - process\_prmtop.c, 92
- bool\_t
  - function\_def.h, 50
- bounds\_check.c
  - calculate\_structure\_diversity, 31
  - check\_angles, 32
  - check\_bonds, 32
  - check\_dihedrals, 32
  - check\_range, 33
  - clean\_up\_bounds, 33
- bounds\_struct, 9
  - angle\_thetas, 9
  - bond\_lengths, 9
  - dihedral\_thetas, 9
  - mem\_allocated, 9
- CREATE\_INPUT
  - function\_def.h, 51

- CHECK\_BOUNDS
  - global\_options\_struct, 16
- CMD\_HELP\_REQ
  - function\_def.h, 48
- CONV\_LIMIT
  - global\_options\_struct, 16
- calc\_angle\_radians
  - function\_def.h, 52
  - misc\_utils.c, 81
- calc\_bond\_length
  - function\_def.h, 53
  - misc\_utils.c, 81
- calc\_dihedral\_radians
  - function\_def.h, 53
  - misc\_utils.c, 81
- calc\_fit\_dimensions
  - function\_def.h, 53
  - misc\_utils.c, 82
- calc\_r\_squared.c
  - calc\_r\_squared\_multiprmtop, 34
- calc\_r\_squared\_multiprmtop
  - calc\_r\_squared.c, 34
  - function\_def.h, 53
- calculate\_no\_fit\_params
  - function\_def.h, 53
  - misc\_utils.c, 82
- calculate\_structure\_diversity
  - bounds\_check.c, 31
  - function\_def.h, 54
- check\_angles
  - bounds\_check.c, 32
  - function\_def.h, 54
- check\_bonds
  - bounds\_check.c, 32
  - function\_def.h, 54
- check\_dihedrals
  - bounds\_check.c, 32
  - function\_def.h, 55
- check\_for\_valid\_filename
  - function\_def.h, 55
  - misc\_utils.c, 82
- check\_range
  - bounds\_check.c, 33
  - function\_def.h, 55
- chrg
  - atom\_struct, 7
- clean\_up\_bounds
  - bounds\_check.c, 33
  - function\_def.h, 56
- cn1
  - parm\_struct, 21
- cn2
  - parm\_struct, 21
- command\_line\_help
  - function\_def.h, 56
  - help\_functions.c, 79
- compare\_energy
  - function\_def.h, 56
- misc\_utils.c, 82
- conduct\_dihedral\_least\_squares
  - dihedral\_fitting.c, 36
  - function\_def.h, 56
- constants.h
  - BOHR\_TO\_ANGSTROM, 34
  - DEGREE\_TO\_RADIAN, 34
  - HARTREE\_TO\_KCALMOL, 34
  - KJMOL\_TO\_KCALMOL, 34
  - PI, 34
  - RADIAN\_TO\_DEGREE, 34
- coord\_set, 9
  - energy\_filename, 10
  - filename, 10
  - mem\_allocated, 10
  - natoms, 10
  - num\_coords, 10
  - struc, 10
- coords\_struct, 10
  - energy, 11
  - force, 11
  - mem\_allocated, 11
  - x\_coord, 11
  - y\_coord, 11
  - z\_coord, 11
- create\_input.c
  - create\_input\_single\_prmtop, 35
  - create\_qm\_input, 35
- create\_input\_single\_prmtop
  - create\_input.c, 35
  - function\_def.h, 56
- create\_qm\_input
  - create\_input.c, 35
  - function\_def.h, 57
- DEFAULT
  - function\_def.h, 51
- DIHEDRAL\_LEAST\_SQUARES
  - function\_def.h, 50
- DATA\_OVERFLOW
  - function\_def.h, 49
- DEBUG
  - function\_def.h, 49
- DEGREE\_TO\_RADIAN
  - constants.h, 34
- DIHEDRAL\_PARAMS
  - global\_options\_struct, 16
- DIHEDRAL\_SPAN
  - global\_options\_struct, 16
- DIHEDRALBH\_dx
  - global\_options\_struct, 16
- DIHEDRALG\_dx
  - global\_options\_struct, 16
- DIHEDRALN\_dx
  - global\_options\_struct, 16
- DIHEDRALS
  - function\_def.h, 49
- DO\_FIT\_KP
  - dihedral\_data\_struct, 12

- DO\_FIT\_KR
  - bond\_data\_struct, 8
- DO\_FIT\_KT
  - angle\_data\_struct, 6
- DO\_FIT\_NP
  - dihedral\_data\_struct, 12
- DO\_FIT\_PHASE
  - dihedral\_data\_struct, 12
- DO\_FIT\_REQ
  - bond\_data\_struct, 8
- DO\_FIT\_THEQ
  - angle\_data\_struct, 6
- defaults.c
  - set\_default\_options, 36
- dihedral\_data
  - parm\_struct, 22
- dihedral\_data\_struct, 11
  - atom1, 12
  - atom2, 12
  - atom3, 12
  - atom4, 12
  - DO\_FIT\_KP, 12
  - DO\_FIT\_NP, 12
  - DO\_FIT\_PHASE, 12
  - number, 12
  - phase, 12
  - pk, 12
  - pn, 12
- dihedral\_fitting.c
  - conduct\_dihedral\_least\_squares, 36
  - dihedral\_least\_squares, 37
- dihedral\_least\_squares
  - dihedral\_fitting.c, 37
  - function\_def.h, 57
- dihedral\_thetas
  - bounds\_struct, 9
- dihedral\_type\_struct, 12
  - atom\_type1, 13
  - atom\_type2, 13
  - atom\_type3, 13
  - atom\_type4, 13
  - improper, 13
  - num\_terms, 13
  - term, 13
- dihedral\_types\_equal
  - function\_def.h, 58
  - misc\_utils.c, 82
- dihedralcomparator
  - function\_def.h, 58
  - process\_prmtop.c, 92
- do\_fit
  - fitting\_control.c, 44
  - function\_def.h, 58
- do\_mutation
  - function\_def.h, 58
  - genetic\_algorithm.c, 78
- double\_2D\_array\_free
  - function\_def.h, 59
- mem\_alloc.c, 80
- elements.c
  - find\_atomic\_number\_from\_parm, 37
  - print\_atomic\_number\_as\_symbol, 37
- energy
  - coords\_struct, 11
- energy\_filename
  - coord\_set, 10
- energy\_t
  - function\_def.h, 50
- error\_messages.c
  - file\_open\_failure, 38
  - malloc\_failure\_char, 38
  - malloc\_failure\_double, 38
  - malloc\_failure\_int, 38
  - malloc\_failure\_short\_int, 39
  - process\_retval, 39
- eval\_amber\_forces.c
  - eval\_amber\_forces\_single\_struct, 40
  - eval\_sum\_amber\_forces, 40
  - eval\_sum\_amber\_forces\_multiprm\_top, 40
  - mark\_relevant\_atoms, 41
  - print\_forces, 41
- eval\_amber\_forces\_single\_struct
  - eval\_amber\_forces.c, 40
  - function\_def.h, 59
- eval\_amber\_std.c
  - eval\_amber\_std\_for\_single\_struct, 42
  - eval\_sum\_squares\_amber\_std, 42
  - eval\_sum\_squares\_amber\_std\_multiprm\_top, 42
- eval\_amber\_std\_for\_single\_struct
  - eval\_amber\_std.c, 42
  - function\_def.h, 59
- eval\_sum\_amber\_forces
  - eval\_amber\_forces.c, 40
  - function\_def.h, 59
- eval\_sum\_amber\_forces\_multiprm\_top
  - eval\_amber\_forces.c, 40
  - function\_def.h, 60
- eval\_sum\_squares\_amber\_std
  - eval\_amber\_std.c, 42
  - function\_def.h, 60
- eval\_sum\_squares\_amber\_std\_multiprm\_top
  - eval\_amber\_std.c, 42
  - function\_def.h, 60
- FALSE
  - function\_def.h, 50
- FIT
  - function\_def.h, 51
- FAILURE
  - function\_def.h, 49
- FILE\_OPEN\_FAIL
  - function\_def.h, 49
- FILE\_READ\_FAIL
  - function\_def.h, 49
- FIT\_PHASE
  - global\_options\_struct, 16

FUNC\_TO\_FIT  
     global\_options\_struct, 16  
 file\_io.c  
     read\_job\_control\_file, 43  
     read\_parameter\_file, 43  
     read\_parameter\_file\_v2, 44  
 file\_open\_failure  
     error\_messages.c, 38  
     function\_def.h, 60  
 filename  
     coord\_set, 10  
     parm\_struct, 22  
 find\_atomic\_number\_from\_parm  
     elements.c, 37  
     function\_def.h, 61  
 find\_flag  
     function\_def.h, 61  
     misc\_utils.c, 83  
 fit\_atom  
     parm\_struct, 22  
 fitting\_control.c  
     do\_fit, 44  
 force  
     coords\_struct, 11  
 force\_struct, 14  
     x, 14  
     y, 14  
     z, 14  
 force\_t  
     function\_def.h, 50  
 free\_coords  
     function\_def.h, 61  
     mem\_alloc.c, 80  
 free\_data\_matrix  
     function\_def.h, 61  
     genetic\_algorithm.c, 78  
 free\_prmtop  
     function\_def.h, 61  
     mem\_alloc.c, 80  
 function\_def.h  
     ADF, 51  
     AMBER\_FORCES, 50  
     BOTH, 50  
     CREATE\_INPUT, 51  
     DEFAULT, 51  
     DIHEDRAL\_LEAST\_SQUARES, 50  
     FALSE, 50  
     FIT, 51  
     GAMESS, 51  
     GAUSSIAN, 51  
     GENETIC, 50  
     HARTREE, 50  
     HARTREE\_BOHR, 50  
     HIGH, 51  
     K\_ONLY, 51  
     KCALMOL, 50  
     KCALMOL\_ANGSTROM, 50  
     KJMOL, 50  
     LOAD, 51  
     LOW, 51  
     MEDIUM, 51  
     NO, 50  
     NONE, 50  
     READ, 51  
     SAVE, 51  
     SET\_PARAMS, 51  
     SIMPLEX, 50  
     SUM\_SQUARES\_AMBER\_STANDARD, 50  
     TRUE, 50  
     WRITE, 51  
     YES, 50  
 function\_def.h  
     \_GNU\_SOURCE, 48  
     ABORT, 48  
     ALLOC\_FAIL, 48  
     ANGLES, 48  
     algorithm\_t, 50  
     alloc\_2D\_double, 52  
     alloc\_coords, 52  
     alloc\_data\_matrix, 52  
     anglecomparator, 52  
     BONDS, 48  
     bondcomparator, 52  
     bool\_t, 50  
     CMD\_HELP\_REQ, 48  
     calc\_angle\_radians, 52  
     calc\_bond\_length, 53  
     calc\_dihedral\_radians, 53  
     calc\_fit\_dimensions, 53  
     calc\_r\_squared\_multiprmtop, 53  
     calculate\_no\_fit\_params, 53  
     calculate\_structure\_diversity, 54  
     check\_angles, 54  
     check\_bonds, 54  
     check\_dihedrals, 55  
     check\_for\_valid\_filename, 55  
     check\_range, 55  
     clean\_up\_bounds, 56  
     command\_line\_help, 56  
     compare\_energy, 56  
     conduct\_dihedral\_least\_squares, 56  
     create\_input\_single\_prmtop, 56  
     create\_qm\_input, 57  
     DATA\_OVERFLOW, 49  
     DEBUG, 49  
     DIHEDRALS, 49  
     dihedral\_least\_squares, 57  
     dihedral\_types\_equal, 58  
     dihedralcomparator, 58  
     do\_fit, 58  
     do\_mutation, 58  
     double\_2D\_array\_free, 59  
     energy\_t, 50  
     eval\_amber\_forces\_single\_struct, 59  
     eval\_amber\_std\_for\_single\_struct, 59  
     eval\_sum\_amber\_forces, 59

- eval\_sum\_amber\_forces\_multiprmtop, 60
- eval\_sum\_squares\_amber\_std, 60
- eval\_sum\_squares\_amber\_std\_multiprmtop, 60
- FAILURE, 49
- FILE\_OPEN\_FAIL, 49
- FILE\_READ\_FAIL, 49
- file\_open\_failure, 60
- find\_atomic\_number\_from\_parm, 61
- find\_flag, 61
- force\_t, 50
- free\_coords, 61
- free\_data\_matrix, 61
- free\_prmtop, 61
- function\_t, 50
- genetic\_wizard, 62
- get\_float, 62
- get\_option, 62
- global\_unlock, 62
- HELP\_REQ, 49
- HIST\_REQ, 49
- handle\_sigint, 62
- INVALID\_DATA, 49
- INVALID\_FORMAT, 49
- INVALID\_LINE, 49
- job\_control\_wizard, 62
- MINSTATIC, 49
- malloc\_failure\_char, 63
- malloc\_failure\_double, 63
- malloc\_failure\_int, 63
- malloc\_failure\_short\_int, 63
- minimise\_function\_genetic, 63
- minimise\_function\_simplex, 64
- modify\_params\_scratch\_data, 64
- NOT\_IMPLEMENTED, 49
- name\_copy, 64
- not\_enough\_dihedrals, 65
- OFF, 49
- ON, 49
- ObfuscateAtom, 65
- parameter\_mode\_t, 50
- print\_atomic\_number\_as\_symbol, 65
- print\_backtrace, 65
- print\_close\_line\_box, 65
- print\_dihedral, 66
- print\_forces, 66
- print\_job\_control\_summary, 66
- print\_multiprmtop\_summary, 66
- print\_open\_line\_box, 67
- print\_parameter\_summary, 67
- print\_program\_history, 67
- print\_program\_info, 67
- process\_command\_line, 67
- process\_job\_control\_setting, 68
- process\_prmtops, 68
- process\_retval, 68
- process\_single\_prmtop, 68
- qm\_format\_t, 51
- RAND\_RATIO, 49
- read\_gaussian\_energy, 69
- read\_gaussian\_forces, 69
- read\_input\_parameters, 69
- read\_job\_control\_file, 70
- read\_mdcrds, 70
- read\_parameter\_file, 70
- read\_prmtops, 71
- read\_qm, 71
- read\_qm\_directory, 71
- read\_qm\_energy\_list, 72
- read\_single\_mdcrd, 72
- read\_single\_prmtop, 72
- readwrite\_t, 51
- runtype\_t, 51
- s\_getline, 73
- SUCCESS, 49
- set\_default\_options, 73
- set\_dihedral\_fit, 73
- simplex\_wizard, 73
- TOO\_MANY\_OPT, 49
- UNKNOWN\_ELEMENT, 49
- UNKNOWN\_OPT, 49
- unObfuscateAtom, 73
- update\_prmtop\_data, 74
- verbosity\_t, 51
- verify\_prmtops, 74
- WARN, 49
- write\_energy, 74
- write\_frmod, 75
- write\_input\_adf, 75
- write\_input\_gamess, 75
- write\_input\_gaussian, 75
- write\_input\_parameters, 76
- write\_prmtop, 76
- function\_t
  - function\_def.h, 50
- GAMESS
  - function\_def.h, 51
- GAUSSIAN
  - function\_def.h, 51
- GENETIC
  - function\_def.h, 50
- genetic\_algorithm.c
  - alloc\_data\_matrix, 77
  - do\_mutation, 78
  - free\_data\_matrix, 78
  - minimise\_function\_genetic, 78
- genetic\_wizard
  - function\_def.h, 62
  - wizard.c, 99
- get\_float
  - function\_def.h, 62
  - wizard.c, 99
- get\_option
  - function\_def.h, 62
  - wizard.c, 99
- global\_options\_struct, 14
  - ALGORITHM, 15

- ANGLE\_LIMIT, 15
- ANGLE\_PARAMS, 15
- ANGLEEQ\_dx, 15
- ANGLEFC\_dx, 15
- BOND\_LIMIT, 16
- BOND\_PARAMS, 16
- BONDEQ\_dx, 16
- BONDFC\_dx, 16
- CHECK\_BOUNDS, 16
- CONV\_LIMIT, 16
- DIHEDRAL\_PARAMS, 16
- DIHEDRAL\_SPAN, 16
- DIHEDRALBH\_dx, 16
- DIHEDRALG\_dx, 16
- DIHEDRALN\_dx, 16
- FIT\_PHASE, 16
- FUNC\_TO\_FIT, 16
- job\_control\_filename, 17
- K, 17
- K\_FIT, 17
- K\_dx, 17
- MAX\_GENERATIONS, 17
- MUTATION\_RATE, 17
- mdcrd\_list, 17
- mem\_allocated, 17
- NDIHEDRALS, 17
- NDIMENSIONS, 17
- NOOPTIMIZATIONS, 17
- num\_prmtops, 18
- PARENT\_PERCENT, 18
- prmtop\_list, 18
- QM\_ENERGY\_UNITS, 18
- QM\_FORCE\_UNITS, 18
- QM\_SYSTEM\_CHARGE, 18
- QMFILEFORMAT, 18
- QMFILEOUTEND, 18
- QMFILEOUTSTART, 18
- QMHEADER, 18
- RANDOM\_SEED, 18
- RUNTYPE, 19
- SCATTERPLOTS, 19
- SCEE, 19
- SCNB, 19
- SEARCH\_SPACE, 19
- TOTAL\_STRUCTURES, 19
- VERBOSITY, 19
- WRITE\_ENERGY, 19
- WRITE\_FRCMOD, 19
- WRITE\_PRMTOP, 19
- global\_unlock
  - function\_def.h, 62
  - mem\_alloc.c, 80
- HARTREE
  - function\_def.h, 50
- HARTREE\_BOHR
  - function\_def.h, 50
- HIGH
  - function\_def.h, 51
- HARTREE\_TO\_KCALMOL
  - constants.h, 34
- HELP\_REQ
  - function\_def.h, 49
- HIST\_REQ
  - function\_def.h, 49
- handle\_sigint
  - function\_def.h, 62
  - misc\_utils.c, 83
- hbcut
  - parm\_struct, 22
- help\_functions.c
  - command\_line\_help, 79
- IFBOX
  - parm\_struct, 22
- IFCAP
  - parm\_struct, 22
- IFPERT
  - parm\_struct, 22
- INVALID\_DATA
  - function\_def.h, 49
- INVALID\_FORMAT
  - function\_def.h, 49
- INVALID\_LINE
  - function\_def.h, 49
- iac
  - atom\_struct, 7
- ib
  - parmbond\_struct, 28
- icb
  - parmbond\_struct, 28
- icp
  - parmdihedral\_struct, 28
- ict
  - parmangle\_struct, 27
- igraph
  - atom\_struct, 7
- improper
  - dihedral\_type\_struct, 13
- ip
  - parmdihedral\_struct, 28
- ipres
  - residue, 29
- irotat
  - atom\_struct, 7
- isymb1
  - atom\_struct, 7
- it
  - parmangle\_struct, 27
- itree
  - atom\_struct, 7
- JHPARM
  - parm\_struct, 22
- JPARM
  - parm\_struct, 22
- jb
  - parmbond\_struct, 28



- job\_control\_filename
  - global\_options\_struct, 17
- job\_control\_wizard
  - function\_def.h, 62
  - wizard.c, 100
- join
  - atom\_struct, 7
- jp
  - parmdihedral\_struct, 29
- jt
  - parmangle\_struct, 27
- K
  - global\_options\_struct, 17
- K\_ONLY
  - function\_def.h, 51
- KCALMOL
  - function\_def.h, 50
- KCALMOL\_ANGSTROM
  - function\_def.h, 50
- KJMOL
  - function\_def.h, 50
- K\_FIT
  - global\_options\_struct, 17
- K\_dx
  - global\_options\_struct, 17
- KJMOL\_TO\_KCALMOL
  - constants.h, 34
- kp
  - parmdihedral\_struct, 29
- kt
  - parmangle\_struct, 27
- LOAD
  - function\_def.h, 51
- LOW
  - function\_def.h, 51
- labres
  - residue, 29
- lp
  - parmdihedral\_struct, 29
- MEDIUM
  - function\_def.h, 51
- MAX\_GENERATIONS
  - global\_options\_struct, 17
- MBONA
  - parm\_struct, 22
- MBPER
  - parm\_struct, 22
- MDPER
  - parm\_struct, 22
- MGPER
  - parm\_struct, 23
- MINSTATIC
  - function\_def.h, 49
- MPHIA
  - parm\_struct, 23
- MPTRA
  - parm\_struct, 23
- MTHETS
  - parm\_struct, 23
- MUMANG
  - parm\_struct, 23
- MUMBND
  - parm\_struct, 23
- MUTATION\_RATE
  - global\_options\_struct, 17
- main
  - parameter\_optimiser.c, 88
- malloc\_failure\_char
  - error\_messages.c, 38
  - function\_def.h, 63
- malloc\_failure\_double
  - error\_messages.c, 38
  - function\_def.h, 63
- malloc\_failure\_int
  - error\_messages.c, 38
  - function\_def.h, 63
- malloc\_failure\_short\_int
  - error\_messages.c, 39
  - function\_def.h, 63
- mark\_relevant\_atoms
  - eval\_amber\_forces.c, 41
- mdcrd\_list
  - global\_options\_struct, 17
- mem\_alloc.c
  - alloc\_2D\_double, 79
  - alloc\_coords, 79
  - double\_2D\_array\_free, 80
  - free\_coords, 80
  - free\_prmtop, 80
  - global\_unlock, 80
- mem\_allocated
  - bounds\_struct, 9
  - coord\_set, 10
  - coords\_struct, 11
  - global\_options\_struct, 17
  - parm\_struct, 23
- minimise\_function\_genetic
  - function\_def.h, 63
  - genetic\_algorithm.c, 78
- minimise\_function\_simplex
  - function\_def.h, 64
  - simplex.c, 98
- misc\_utils.c
  - calc\_angle\_radians, 81
  - calc\_bond\_length, 81
  - calc\_dihedral\_radians, 81
  - calc\_fit\_dimensions, 82
  - calculate\_no\_fit\_params, 82
  - check\_for\_valid\_filename, 82
  - compare\_energy, 82
  - dihedral\_types\_equal, 82
  - find\_flag, 83
  - handle\_sigint, 83
  - modify\_params\_scratch\_data, 83

- name\_copy, 83
- not\_enough\_dihedrals, 84
- ObfuscateAtom, 84
- print\_backtrace, 84
- print\_close\_line\_box, 84
- print\_dihedral, 85
- print\_open\_line\_box, 85
- s\_getline, 85
- unObfuscateAtom, 85
- update\_prmtop\_data, 86
- modify\_params\_scratch\_data
  - function\_def.h, 64
  - misc\_utils.c, 83
- NO
  - function\_def.h, 50
- NONE
  - function\_def.h, 50
- NAME\_DEFAULT
  - prmtop\_params.h, 89
- NAME\_SIZE
  - prmtop\_params.h, 89
- NATYP
  - parm\_struct, 23
- NBONA
  - parm\_struct, 23
- NBONH
  - parm\_struct, 23
- NBPER
  - parm\_struct, 23
- NDIHEDRALS
  - global\_options\_struct, 17
- NDIMENSIONS
  - global\_options\_struct, 17
- NDPER
  - parm\_struct, 24
- NEXT
  - parm\_struct, 24
- NGPER
  - parm\_struct, 24
- NHB
  - parm\_struct, 24
- NMXRS
  - parm\_struct, 24
- NOPTIMIZATIONS
  - global\_options\_struct, 17
- NOT\_IMPLEMENTED
  - function\_def.h, 49
- NPHIA
  - parm\_struct, 24
- NPHIH
  - parm\_struct, 24
- NSIMPLEX\_OUTER\_MAX
  - function\_def.h, 49
- NTHETA
  - parm\_struct, 24
- NTHETH
  - parm\_struct, 24
- NTOTAT
  - parm\_struct, 25
- NTOTRS
  - parm\_struct, 25
- NTYPES
  - parm\_struct, 25
- NUMEXTRA
  - parm\_struct, 25
- Name
  - prmtop\_params.h, 89
- name\_copy
  - function\_def.h, 64
  - misc\_utils.c, 83
- natex
  - parm\_struct, 23
- natoms
  - coord\_set, 10
- ndimensions
  - parm\_struct, 24
- newparm
  - parm\_struct, 24
- nno
  - parm\_struct, 24
- not\_enough\_dihedrals
  - function\_def.h, 65
  - misc\_utils.c, 84
- num\_coords
  - coord\_set, 10
- num\_prmtops
  - global\_options\_struct, 18
- num\_terms
  - dihedral\_type\_struct, 13
- number
  - angle\_data\_struct, 6
  - bond\_data\_struct, 8
  - dihedral\_data\_struct, 12
- numex
  - atom\_struct, 7
- OFF
  - function\_def.h, 49
- ON
  - function\_def.h, 49
- ObfuscateAtom
  - function\_def.h, 65
  - misc\_utils.c, 84
- options\_summary.c
  - print\_job\_control\_summary, 86
- PARAMETERS\_TO\_FIT
  - global\_options\_struct, 18
- PARENT\_PERCENT
  - global\_options\_struct, 18
- PI
  - constants.h, 34
- pangle
  - parm\_struct, 25
- pangleH
  - parm\_struct, 25
- param\_summary.c

- print\_multiprmtop\_summary, [87](#)
  - print\_parameter\_summary, [87](#)
- parameter\_mode\_t
  - function\_def.h, [50](#)
- parameter\_optimiser.c
  - main, [88](#)
- parm\_struct, [19](#)
  - ag, [21](#)
  - angle\_data, [21](#)
  - atom, [21](#)
  - bg, [21](#)
  - bond\_data, [21](#)
  - cn1, [21](#)
  - cn2, [21](#)
  - dihedral\_data, [22](#)
  - filename, [22](#)
  - fit\_atom, [22](#)
  - hbcut, [22](#)
  - IFBOX, [22](#)
  - IFCAP, [22](#)
  - IFPERT, [22](#)
  - JHPARM, [22](#)
  - JPARM, [22](#)
  - MBONA, [22](#)
  - MBPER, [22](#)
  - MDPER, [22](#)
  - MGPER, [23](#)
  - MPHIA, [23](#)
  - MPTRA, [23](#)
  - MTHETS, [23](#)
  - MUMANG, [23](#)
  - MUMBND, [23](#)
  - mem\_allocated, [23](#)
  - NATYP, [23](#)
  - NBONA, [23](#)
  - NBONH, [23](#)
  - NBPER, [23](#)
  - NDPER, [24](#)
  - NEXT, [24](#)
  - NGPER, [24](#)
  - NHB, [24](#)
  - NMXRS, [24](#)
  - NPHIA, [24](#)
  - NPHIH, [24](#)
  - NTHETA, [24](#)
  - NTHETH, [24](#)
  - NTOTAT, [25](#)
  - NTOTRS, [25](#)
  - NTYPES, [25](#)
  - NUMEXTRA, [25](#)
  - natex, [23](#)
  - ndimensions, [24](#)
  - newparm, [24](#)
  - nno, [24](#)
  - pangle, [25](#)
  - pangleH, [25](#)
  - pbond, [25](#)
  - pbondH, [25](#)
  - pdihedral, [25](#)
  - pdihedralH, [25](#)
  - phase, [25](#)
  - pk, [25](#)
  - pn, [26](#)
  - req, [26](#)
  - residue, [26](#)
  - rk, [26](#)
  - solty, [26](#)
  - teq, [26](#)
  - title, [26](#)
  - tk, [26](#)
  - unique\_angles\_found, [26](#)
  - unique\_bonds\_found, [26](#)
  - unique\_dihedral\_terms, [26](#)
  - unique\_dihedrals\_found, [26](#)
- parmangle\_struct, [27](#)
  - ict, [27](#)
  - it, [27](#)
  - jt, [27](#)
  - kt, [27](#)
- parmbond\_struct, [27](#)
  - ib, [28](#)
  - icb, [28](#)
  - jb, [28](#)
- parmdihedral\_struct, [28](#)
  - icp, [28](#)
  - ip, [28](#)
  - jp, [29](#)
  - kp, [29](#)
  - lp, [29](#)
- pbond
  - parm\_struct, [25](#)
- pbondH
  - parm\_struct, [25](#)
- pdihedral
  - parm\_struct, [25](#)
- pdihedralH
  - parm\_struct, [25](#)
- phase
  - dihedral\_data\_struct, [12](#)
  - parm\_struct, [25](#)
- pk
  - dihedral\_data\_struct, [12](#)
  - parm\_struct, [25](#)
- pn
  - dihedral\_data\_struct, [12](#)
  - parm\_struct, [26](#)
- print\_atomic\_number\_as\_symbol
  - elements.c, [37](#)
  - function\_def.h, [65](#)
- print\_backtrace
  - function\_def.h, [65](#)
  - misc\_utils.c, [84](#)
- print\_close\_line\_box
  - function\_def.h, [65](#)
  - misc\_utils.c, [84](#)
- print\_dihedral

- function\_def.h, 66
- misc\_utils.c, 85
- print\_forces
  - eval\_amber\_forces.c, 41
  - function\_def.h, 66
- print\_job\_control\_summary
  - function\_def.h, 66
  - options\_summary.c, 86
- print\_multiprmtop\_summary
  - function\_def.h, 66
  - param\_summary.c, 87
- print\_open\_line\_box
  - function\_def.h, 67
  - misc\_utils.c, 85
- print\_parameter\_summary
  - function\_def.h, 67
  - param\_summary.c, 87
- print\_program\_history
  - function\_def.h, 67
  - print\_program\_info.c, 88
- print\_program\_info
  - function\_def.h, 67
  - print\_program\_info.c, 88
- print\_program\_info.c
  - print\_program\_history, 88
  - print\_program\_info, 88
- prmtop\_list
  - global\_options\_struct, 18
- prmtop\_params.h
  - BUFFER\_SIZE, 89
  - NAME\_DEFAULT, 89
  - NAME\_SIZE, 89
  - Name, 89
- process\_command\_line
  - function\_def.h, 67
  - process\_command\_line.c, 90
- process\_command\_line.c
  - process\_command\_line, 90
- process\_job\_control\_setting
  - function\_def.h, 68
  - process\_job\_control\_setting.c, 91
- process\_job\_control\_setting.c
  - process\_job\_control\_setting, 91
- process\_prmtop.c
  - anglecomparator, 92
  - bondcomparator, 92
  - dihedralcomparator, 92
  - process\_prmtops, 92
  - process\_single\_prmtop, 93
  - set\_dihedral\_fit, 93
  - verify\_prmtops, 93
- process\_prmtops
  - function\_def.h, 68
  - process\_prmtop.c, 92
- process\_retval
  - error\_messages.c, 39
  - function\_def.h, 68
- process\_single\_prmtop
  - function\_def.h, 68
  - process\_prmtop.c, 93
- QM\_ENERGY\_UNITS
  - global\_options\_struct, 18
- QM\_FORCE\_UNITS
  - global\_options\_struct, 18
- QM\_SYSTEM\_CHARGE
  - global\_options\_struct, 18
- QMFILEFORMAT
  - global\_options\_struct, 18
- QMFILEOUTEND
  - global\_options\_struct, 18
- QMFILEOUTSTART
  - global\_options\_struct, 18
- QMHEADER
  - global\_options\_struct, 18
- qm\_format\_t
  - function\_def.h, 51
- READ
  - function\_def.h, 51
- RADIAN\_TO\_DEGREE
  - constants.h, 34
- RAND\_RATIO
  - function\_def.h, 49
- RANDOM\_SEED
  - global\_options\_struct, 18
- RUNTYPE
  - global\_options\_struct, 19
- read\_energy.c
  - read\_gaussian\_energy, 94
  - read\_gaussian\_forces, 94
  - read\_qm, 95
  - read\_qm\_directory, 95
  - read\_qm\_energy\_list, 96
- read\_gaussian\_energy
  - function\_def.h, 69
  - read\_energy.c, 94
- read\_gaussian\_forces
  - function\_def.h, 69
  - read\_energy.c, 94
- read\_input\_parameters
  - function\_def.h, 69
- read\_job\_control\_file
  - file\_io.c, 43
  - function\_def.h, 70
- read\_mdcrd.c
  - read\_mdcrds, 96
  - read\_single\_mdcrd, 97
- read\_mdcrds
  - function\_def.h, 70
  - read\_mdcrd.c, 96
- read\_parameter\_file
  - file\_io.c, 43
  - function\_def.h, 70
- read\_parameter\_file\_v2
  - file\_io.c, 44
- read\_prmtop.c

- read\_prmtops, 97
- read\_single\_prmtop, 98
- read\_prmtops
  - function\_def.h, 71
  - read\_prmtop.c, 97
- read\_qm
  - function\_def.h, 71
  - read\_energy.c, 95
- read\_qm\_directory
  - function\_def.h, 71
  - read\_energy.c, 95
- read\_qm\_energy\_list
  - function\_def.h, 72
  - read\_energy.c, 96
- read\_single\_mdcrd
  - function\_def.h, 72
  - read\_mdcrd.c, 97
- read\_single\_prmtop
  - function\_def.h, 72
  - read\_prmtop.c, 98
- readwrite\_t
  - function\_def.h, 51
- req
  - bond\_data\_struct, 8
  - parm\_struct, 26
- res
  - atom\_struct, 7
- residue, 29
  - ipres, 29
  - labres, 29
  - parm\_struct, 26
- rk
  - bond\_data\_struct, 8
  - parm\_struct, 26
- runtype\_t
  - function\_def.h, 51
- SAVE
  - function\_def.h, 51
- SET\_PARAMS
  - function\_def.h, 51
- SIMPLEX
  - function\_def.h, 50
- SUM\_SQUARES\_AMBER\_STANDARD
  - function\_def.h, 50
- s\_getline
  - function\_def.h, 73
  - misc\_utils.c, 85
- SCATTERPLOTS
  - global\_options\_struct, 19
- SCEE
  - global\_options\_struct, 19
- SCNB
  - global\_options\_struct, 19
- SEARCH\_SPACE
  - global\_options\_struct, 19
- SUCCESS
  - function\_def.h, 49
- set\_default\_options
  - defaults.c, 36
  - function\_def.h, 73
- set\_dihedral\_fit
  - function\_def.h, 73
  - process\_prmtop.c, 93
- simplex.c
  - minimise\_function\_simplex, 98
- simplex\_wizard
  - function\_def.h, 73
  - wizard.c, 100
- solty
  - parm\_struct, 26
- struc
  - coord\_set, 10
- TRUE
  - function\_def.h, 50
- TOO\_MANY\_OPT
  - function\_def.h, 49
- TOTAL\_STRUCTURES
  - global\_options\_struct, 19
- teq
  - angle\_data\_struct, 6
  - parm\_struct, 26
- term
  - dihedral\_type\_struct, 13
- title
  - parm\_struct, 26
- tk
  - angle\_data\_struct, 6
  - parm\_struct, 26
- UNKNOWN\_ELEMENT
  - function\_def.h, 49
- UNKNOWN\_OPT
  - function\_def.h, 49
- unObfuscateAtom
  - function\_def.h, 73
  - misc\_utils.c, 85
- unique\_angles\_found
  - parm\_struct, 26
- unique\_bonds\_found
  - parm\_struct, 26
- unique\_dihedral\_terms
  - parm\_struct, 26
- unique\_dihedrals\_found
  - parm\_struct, 26
- update\_prmtop\_data
  - function\_def.h, 74
  - misc\_utils.c, 86
- VERBOSITY
  - global\_options\_struct, 19
- verbosity\_t
  - function\_def.h, 51
- verify\_prmtops
  - function\_def.h, 74
  - process\_prmtop.c, 93

WRITE  
    function\_def.h, 51  
WARN  
    function\_def.h, 49  
WRITE\_ENERGY  
    global\_options\_struct, 19  
WRITE\_FRCMOD  
    global\_options\_struct, 19  
WRITE\_PRMTOP  
    global\_options\_struct, 19  
wizard.c  
    genetic\_wizard, 99  
    get\_float, 99  
    get\_option, 99  
    job\_control\_wizard, 100  
    simplex\_wizard, 100  
write\_energy  
    function\_def.h, 74  
    write\_input.c, 101  
write\_frcmod  
    function\_def.h, 75  
    write\_input.c, 101  
write\_input.c  
    write\_energy, 101  
    write\_frcmod, 101  
    write\_input\_adf, 101  
    write\_input\_gamess, 101  
    write\_input\_gaussian, 102  
    write\_input\_parameters, 102  
    write\_prmtop, 102  
write\_input\_adf  
    function\_def.h, 75  
    write\_input.c, 101  
write\_input\_gamess  
    function\_def.h, 75  
    write\_input.c, 101  
write\_input\_gaussian  
    function\_def.h, 75  
    write\_input.c, 102  
write\_input\_parameters  
    function\_def.h, 76  
    write\_input.c, 102  
write\_prmtop  
    function\_def.h, 76  
    write\_input.c, 102  
  
x  
    force\_struct, 14  
x\_coord  
    coords\_struct, 11  
  
y  
    force\_struct, 14  
YES  
    function\_def.h, 50  
y\_coord  
    coords\_struct, 11  
  
z  
    force\_struct, 14  
z\_coord  
    coords\_struct, 11