Paramfit

Generated by Doxygen 1.8.2

Mon Jun 24 2013 17:11:23

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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 isymbl
- 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::isymbl

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/prmtop_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 Vn/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_MULTIPLICITY
- 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 Vn/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::NOPTIMIZATIONS

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 runtype_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 ffrcmod 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 >=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 uniquness 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	*global_options*	The global options structure
out	*bounds_data*	The data structure with the data about input structure distributions
in	*parm_data*	Pointer to a single parameter data structure
in	*coords_data*	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

in	*global_options*	The global options structure
in	*parm_data*	Pointer to the parameter data structure, including the angle value to check
in	*coords_data*	Pointer to the coordinate set for these parameters
in	*bounds_data*	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

in	*global_options*	The global options structure
in	*parm_data*	Pointer to the parameter structure with the bond parameters to check inside
in	*coords_data*	Pointer to coordinate structure that bounds data was gathered from
in	*bounds_data*	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	*global_options*	The global options structure
in	*parm_data*	Pointer to single parameter struture containing the dihedral parameters
in	*coords_data*	Pointer to single coordinate set
in	*bounds_data*	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	*global_options*	The global options structure
in,out	*parm_data*	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	*bounds_data*	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_datas, 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 3.14159265358979323846264338327 9 /*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 3.14159265358979323846264338327 9 /*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	*global_options*	The global options structure
in	*parm_data*	Pointer to a single set of parameters for this molecule
in	*coords_data*	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 dircctory, 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	*global_options*	The global options structure
in	*parm_datas*	Pointer to the array of parm structures
in	*coords_datas*	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

in,out	*global_options*	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

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

Returns

Integer indicating success or failure

4.6.2.2 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

in	*global_options*	The global options structure
in,out	*parm_data*	Array containing the dihedral and parameter information at the beginning, and will be updated with the new parameters at the end.
in	*coords_data*	Pointer to single structure containing coordinates and QM energy of each input structure
in	*fit_phase*	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	*routine*	The function where the failure occured
in	*var_name*	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	*routine*	The function where the failure occured
in	*var_name*	The variable that failed to be allocated
in	*chars_requested*	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	*routine*	The function where the failure occured
in	*var_name*	The variable that failed to be allocated
in	*doubles_requested*	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	*routine*	The function where the failure occured
in	*var_name*	The variable that failed to be allocated
in	*ints_requested*	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	*routine*	The function where the failure occured
in	*var_name*	The variable that failed to be allocated
in	*short_ints_requested*	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	*err_code*	The return value from the function
in	*VERBOSITY*	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_datas, coord_set *coords_datas)
- 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	*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

4.9.2.2 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	*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.

4.9.2.3 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	*global_options*	The global options structure
in	*parm_datas*	Array of parameter sets, one for each molecule
in	*coords_datas*	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	*global_options*	The global options structure
in,out	*parm_data*	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	*global_options*	The global options structure
in	*parm_data*	The parameter set to use in the function
in	*coords_data*	Pointer to single set of atom coordinates in input structure
in	*atom*	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_multiprmtop (global_options_struct *global_options, parm_struct *parm_datas, coord_set *coords_datas)
- 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	*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

4.10.2.2 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	*global_options*	The global options structure
in	*parm_data*	The parameter file containing parameter set to be scored
in	*coords_data*	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	*global_options*	The global options structure
in	*parm_datas*	The array of parm structures
in	*coords_datas*	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_datas)
- 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

in,out	*global_options*	The global options structure, which contains the location of the job control file.
in,out	*coords_data*	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 nwe format parameter file and call the v2 function to read it if found.

See Also

read_parameter_file_v2

Parameters

in	*global_options*	The global options structure, containing the file name
in,out	*parm_data*	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_datas*)

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

Parameters

in	*global_options*	The global options structure, containing the file name
in,out	*parm_data*	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_datas, coord_set *coords_datas)

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_datas*, coord_set * *coords_datas*)

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	*global_options*	The global options structure containing user defined options
in,out	*parm_data*	Array of initial parameters and what to fit for each molecule, will have final parameters inserted.
in	*coords_data*	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 ncolumns)
- 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_datas, 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_datas, coord_set *coords_datas)
- 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_datas, coord_set *coords_datas)
- int process_command_line (int argc, char **argv, global_options_struct *global_options, parm_struct **parm_datas, coord_set **coords_datas)
- void command_line_help (char *cmd_line_options[])
- int read_qm (global_options_struct *global_options, parm_struct *parm_datas, coord_set *coords_datas)
- 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_datas, coord_set *coords_datas)
- 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_datas)
- int read_single_prmtop (global_options_struct *global_options, parm_struct *parm_data)
- int process_prmtops (global_options_struct *global_options, parm_struct *parm_datas)
- 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 EXCEEDEDMAXITERATIONS -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 MINSTATIC -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

in	*global_options*	The global options structure
in,out	*c*	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

in	*rows*	The number of rows to be in the array
in	*cols*	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-dimenstional 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-dimenstional 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:

```
  A       D
   \     /
    B--C
*
```

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->B b = B->C c = C->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:

```
  | a(x) a(y) a(z) |     | a(x) a(y) a(z) |
  | b(x) b(y) b(z) | -> | b(x) b(y) b(z) |
  | c(x) c(y) c(z) |     | c(x) c(y) c(z) |
*
```

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

in,out	*global_options*	The global options structure, where NDIMENSIONS is to be updated
in	*parm_datas*	Array of parameter data structures

4.14.3.10 double calc_r_squared_multiprmtop (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	*parm_data*	The parameter structure to examine
in	*MODE*	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	*global_options*	The global options structure
out	*bounds_data*	The data structure with the data about input structure distributions
in	*parm_data*	Pointer to a single parameter data structure
in	*coords_data*	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	*global_options*	The global options structure
in	*parm_data*	Pointer to the parameter data structure, including the angle value to check
in	*coords_data*	Pointer to the coordinate set for these parameters
in	*bounds_data*	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

in	*global_options*	The global options structure
in	*parm_data*	Pointer to the parameter structure with the bond parameters to check inside
in	*coords_data*	Pointer to coordinate structure that bounds data was gathered from
in	*bounds_data*	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

in	*global_options*	The global options structure
in	*parm_data*	Pointer to single parameter struture containing the dihedral parameters
in	*coords_data*	Pointer to single coordinate set
in	*bounds_data*	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

in	*data_string*	The string to check
in	*length*	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	*global_options*	The global options structure
in,out	*parm_data*	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	*bounds_data*	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	*global_options*	The global options structure
in,out	*parm_data*	Array of input parameter structures, will be updated
in	*coords_data*	The array of input coordinate sets with associated QM energies
in	*fit_phase*	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

in	*global_options*	The global options structure
in	*parm_data*	Pointer to a single set of parameters for this molecule
in	*coords_data*	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 dircctory, 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	*global_options*	The global options structure
in	*parm_datas*	Pointer to the array of parm structures
in	*coords_datas*	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

in	*global_options*	The global options structure
in,out	*parm_data*	Array containing the dihedral and parameter information at the beginning, and will be updated with the new parameters at the end.
in	*coords_data*	Pointer to single structure containing coordinates and QM energy of each input structure
in	*fit_phase*	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	*first*	Pointer to first dihedral_type_struct to examine
in	*second*	Pointer to dihedral_type_struct 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_datas*, coord_set * *coords_datas*)

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	*global_options*	The global options structure containing user defined options
in,out	*parm_data*	Array of initial parameters and what to fit for each molecule, will have final parameters inserted.
in	*coords_data*	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	*global_options*	The global options structure
in	*parm_data*	The parameter structure, used to look up what kind of parameter element to mutate is.
in,out	*row*	Array representing a row in the data matrix where mutation will happen
in	*col*	Index in the row to mutate
in	*do_mutate*	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	*global_options*	The global options structure
in	*parm_datas*	Array of parameter sets, one for each molecule
in	*coords_datas*	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	*global_options*	The global options structure
in	*parm_data*	The parameter file containing parameter set to be scored
in	*coords_data*	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	*global_options*	The global options structure
in	*parm_datas*	The array of parm structures
in	*coords_datas*	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	*routine*	The function where the failure occured
in	*var_name*	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 selectected label. Rewinds the file pointer before searching.

Parameters

in,out	*fptr*	The file pointer to use
in	*label*	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	*global_options*	The global options structure
in,out	*c*	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	*dm*	Pointer to the 2D array to be freed
in	*rows*	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	*global_options*	Structure where options will be set
in,out	*file*	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	*min*	The minimimum allowed value, exclusive
in	*max*	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	*signal*	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	*global_options*	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	*routine*	The function where the failure occured
in	*var_name*	The variable that failed to be allocated
in	*chars_requested*	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	*routine*	The function where the failure occured
in	*var_name*	The variable that failed to be allocated
in	*doubles_requested*	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	*routine*	The function where the failure occured
in	*var_name*	The variable that failed to be allocated
in	*ints_requested*	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	*routine*	The function where the failure occured
in	*var_name*	The variable that failed to be allocated
in	*short_ints_requested*	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 algorith minimization.

Parameters

in	*global_options*	The global options structure, with algorithm options inside
in,out	*parm_datas*	Array containing the parameters for each molecule, will be updated each generation
in	*coords_data*	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	*global_options*	The global options structure
in,out	*parm_data*	Array of parameter structures, updated at each iteration
in	*coords_data*	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	*global_options*	The global options structure
in,out	*parm_data*	The parameters data structure to read or write to
in,out	*parameters*	Pre-allocated array size NDIMENSIONS to read or write to
in	*MODE*	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

in,out	*fptr*	File pointer to read from
in,out	*stringp*	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

in	*at*	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

in	*signal*	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

in	*no_spaces*	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

in	*type*	Pointer to the dihedral_type_struct to print
in	*term*	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

in	*global_options*	The global options structure
in	*parm_data*	The parameter set to use in the function
in	*coords_data*	Pointer to single set of atom coordinates in input structure
in	*atom*	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

in	*global_options*	The global options structure
in	*parm_data*	Array of parameter files
in	*coords_datas*	Array of coodinate 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	*global_options*	The global options structure
in	*parm_datas*	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*	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

global_options	The global options structure
parm_data	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_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	*argv*	Pointer to array with all options, argv[0] is the name of the program (paramfit)
in,out	*global_options*	Global options structure that will be updated
in,out	*parm_datas*	Unallocated array of parm structs, will be updated with initial blank one if -p specified
in,out	*coords_datas*	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	*setting_line*	The raw line from the job control file as a character array
in	*length*	The length of this line
in,out	*number_settings*	The number of settings that have been set. Will be updated if successful.
in,out	*global_options*	The global options structure to input settings to
in,out	*coords_data*	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_datas*)

Processes all the parmtop data. Processes each individual parm_struct.

Parameters

in	*global_options*	The global options structure, specifies number of prmtops
in,out	*parm_datas*	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	*err_code*	The return value from the function
in	*VERBOSITY*	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	*global_options*	The global options structure. unique_[bonds,angles,dihedrals]_found will be updated
in,out	*parm_data*	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	*global_options*	The global options structure
in	*parm_data*	Pointer to single parm struct
in,out	*coords_data*	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	*global_options*	The global options structure
in	*parm_data*	Pointer to single parameter file
in,out	*coords_data*	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	*global_options*	The global options structure, which contains the location of the job control file.
in,out	*coords_data*	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_datas*, coord_set ** *s_datas*)

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

Parameters

in	*global_options*	The global options structure
in	*parm_datas*	The array of parameter sets to use
out	*struc_datas*	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

4.14.3.79 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	*global_options*	The global options structure, containing the file name
in,out	*parm_data*	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_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	*global_options*	The global options structure, with the parm list filename
in,out	*parm_data*	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_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	*global_options*	The global options structure, which specifies the data format and energy or forces
in	*parm_datas*	Array of parameter sets corresponding to coordinates to read in
in,out	*coords_datas*	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	*global_options*	The global options structure
in,out	*coords_data*	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

in	*global_options*	The global options structure, containing filename with energies
in	*parm_data*	Pointer to single parameter structure corresponding to this coordinate set
in,out	*coords_data*	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

in,out	*coords_data*	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

in	*global_options*	The global options structure
in,out	*parm_data*	The parameter structure that will have info put in
in	*parm_data->filename*	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	*line*	String that will contain the read line
in	*max*	The maximum number of characters to read
in	*fp*	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	*global_options*	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	*global_options*	The global options structure with prompting options
in,out	*s*	The dihedral type to set options for
in	*t*	Integer indicating the term to fit
in	*d_kp*	Whether to prompt to fit dihedral KP
in	*d_np*	Whether to prompt to fit dihedral NP
in	*d_phase*	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	*global_options*	Structure where options will be set
in,out	*file*	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

in	*at*	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_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	*global_options*	The global options structure
in,out	*parm_data*	Array of parameter data structures that will be updated.
in	*parameters*	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_datas*)

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	*global_options*	The global options structure
in	*parm_datas*	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

in	*global_options*	The global options structure, with filename to save to
in	*parm_data*	Pointer to single parameter set to use when calculating amber energies
in	*coords_data*	Pointer to single coordinate set with attached quantum energy
in	*generation*	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	*global_options*	The global options structure, containing filename to write
in	*parm_data*	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	*global_options*	The global options structure
in	*parm_data*	Pointer to the parameter structure that describes these coordinates
in	*current_struct*	Pointer to the coordinate set that will be used
in	*num*	Number of the structure in teh coordinate set to write
in,out	*fptr*	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	*global_options*	The global options structure
in	*parm_data*	The parameter structure
in	*current_structure*	Pointer to coordinate set to write from
in	*num*	The number of the structure in the coordinate set that will be written
in,out	*fptr*	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	*global_options*	The global options structure
in	*parm_data*	The parameter file
in	*current_struct*	Pointer to coordinate set to use
in	*num*	The number of the structure to write from the coordinate set
in,out	*fptr*	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	*global_options*	Global options structure, containing filename to save ase
in	*parm_data*	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	*global_options*	The global options structure, including the file to save to
in	*parm_data*	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	*rows*	The number of rows to be in the array
in	*cols*	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	*global_options*	The global options structure
in	*parm_data*	The parameter structure, used to look up what kind of parameter element to mutate is.
in,out	*row*	Array representing a row in the data matrix where mutation will happen
in	*col*	Index in the row to mutate
in	*do_mutate*	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	*dm*	Pointer to the 2D array to be freed
in	*rows*	The number of rows in the array

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

Conducts the majority of the genetic algorith minimization.

Parameters

in	*global_options*	The global options structure, with algorithm options inside
in,out	*parm_datas*	Array containing the parameters for each molecule, will be updated each generation
in	*coords_data*	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	*global_options*	The global options structure
in,out	*c*	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	*global_options*	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_prmtop (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 *fptr, char *label)
- int name_copy (FILE *fptr, 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_prmtop_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-dimenstional 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-dimenstional 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:

```
  A       D
   \     /
    B--C
*
```

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->B b = B->C c = C->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:

```
  | a(x) a(y) a(z) |     | a(x) a(y) a(z) |
  | b(x) b(y) b(z) | -> | b(x) b(y) b(z) |
  | c(x) c(y) c(z) |     | c(x) c(y) c(z) |
*
```

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_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

in,out	*global_options*	The global options structure, where NDIMENSIONS is to be updated
in	*parm_datas*	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	*parm_data*	The parameter structure to examine
in	*MODE*	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	*data_string*	The string to check
in	*length*	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	*first*	Pointer to first dihedral_type_struct to examine
in	*second*	Pointer to dihedral_type_struct 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 selectected label. Rewinds the file pointer before searching.

Parameters

in,out	*fptr*	The file pointer to use
in	*label*	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	*signal*	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 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	*global_options*	The global options structure
in,out	*parm_data*	The parameters data structure to read or write to
in,out	*parameters*	Pre-allocated array size NDIMENSIONS to read or write to
in	*MODE*	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

in,out	*fptr*	File pointer to read from
in,out	*stringp*	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

in	*at*	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

in	*signal*	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

in	*no_spaces*	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

in	*type*	Pointer to the dihedral_type_struct to print
in	*term*	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

in,out	*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

in,out	*line*	String that will contain the read line
in	*max*	The maximum number of characters to read
in	*fp*	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	*global_options*	The global options structure
in,out	*parm_data*	Array of parameter data structures that will be updated.
in	*parameters*	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	*global_options*	The global options structure
in	*parm_data*	Array of parameter files
in	*coords_datas*	Array of coodinate 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	*global_options*	The global options structure
in	*parm_datas*	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

global_options	The global options structure
parm_data	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	*argv*	Pointer to array with all options, argv[0] is the name of the program (paramfit)
in,out	*global_options*	Global options structure that will be updated
in,out	*parm_datas*	Unallocated array of parm structs, will be updated with initial blank one if -p specified
in,out	*coords_datas*	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	*setting_line*	The raw line from the job control file as a character array
in	*length*	The length of this line
in,out	*number_settings*	The number of settings that have been set. Will be updated if successful.
in,out	*global_options*	The global options structure to input settings to
in,out	*coords_data*	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 seperate parameters for each bond, angle and dihedral type.

This is necessary because the prmtop file and thus prmtop storage arrays do not keep seperate 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_datas*)

Processes all the parmtop data. Processes each individual parm_struct.

Parameters

in	*global_options*	The global options structure, specifies number of prmtops
in,out	*parm_datas*	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

in,out	*global_options*	The global options structure. unique_[bonds,angles,dihedrals]_found will be updated
in,out	*parm_data*	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

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

4.26.2.7 int verify_prmtops (global_options_struct * *global_options*, parm_struct * *parm_datas*)

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	*global_options*	The global options structure
in	*parm_datas*	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	*global_options*	The global options structure
in	*parm_data*	Pointer to single parm struct
in,out	*coords_data*	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	*global_options*	The global options structure
in	*parm_data*	Pointer to single parameter file
in,out	*coords_data*	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	*global_options*	The global options structure, which specifies the data format and energy or forces
in	*parm_datas*	Array of parameter sets corresponding to coordinates to read in
in,out	*coords_datas*	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	*global_options*	The global options structure
in,out	*coords_data*	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	*global_options*	The global options structure, containing filename with energies
in	*parm_data*	Pointer to single parameter structure corresponding to this coordinate set
in,out	*coords_data*	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	*global_options*	The global options structure
in	*parm_datas*	The array of parameter sets to use
out	*struc_datas*	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

in,out	*coords_data*	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	*global_options*	The global options structure, with the parm list filename
in,out	*parm_data*	Pointer to an array of parm_structs, will be reallocd.

Returns

Integer indicating success or failure

4.29.2.2 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

in	*global_options*	The global options structure
in,out	*parm_data*	The parameter structure that will have info put in
in	*parm_data->filename*	The path to the prmtop 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	*global_options*	The global options structure
in,out	*parm_data*	Array of parameter structures, updated at each iteration
in	*coords_data*	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	*global_options*	Structure where options will be set
in,out	*file*	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	*min*	The minimimum allowed value, exclusive
in	*max*	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

in,out	*global_options*	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

in,out	*global_options*	Structure where options will be set
in,out	*file*	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_frcmod (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_prmtop (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	*global_options*	The global options structure, with filename to save to
in	*parm_data*	Pointer to single parameter set to use when calculating amber energies
in	*coords_data*	Pointer to single coordinate set with attached quantum energy
in	*generation*	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	*global_options*	The global options structure, containing filename to write
in	*parm_data*	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	*global_options*	The global options structure
in	*parm_data*	Pointer to the parameter structure that describes these coordinates
in	*current_struct*	Pointer to the coordinate set that will be used
in	*num*	Number of the structure in teh coordinate set to write
in,out	*fptr*	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	*global_options*	The global options structure
in	*parm_data*	The parameter structure
in	*current_structure*	Pointer to coordinate set to write from
in	*num*	The number of the structure in the coordinate set that will be written
in,out	*fptr*	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	*global_options*	The global options structure
in	*parm_data*	The parameter file
in	*current_struct*	Pointer to coordinate set to use
in	*num*	The number of the structure to write from the coordinate set
in,out	*fptr*	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	*global_options*	Global options structure, containing filename to save ase
in	*parm_data*	The parameter file containing which parameters are to be fit

Returns

Integer indicating success or failure

4.32.2.7 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	*global_options*	The global options structure, including the file to save to
in	*parm_data*	The parameters to put into the prmtop file

Returns

Integer indicating success or failure

Index