CBEMD: Parallelized MD in Various Thermodynamic Ensembles

Generated by Doxygen 1.8.2

Tue Jan 8 2013 00:17:40

Contents

1 Hierarchical Index 1

1.1 Class Hierarchy 1

2 Class Index 3

2.1 Class List 3

3 File Index 5

3.1 File List 5

4 Class Documentation 7

4.1 Andersen Class Reference 7

4.1.1 Detailed Description 7

4.1.2 Constructor & Destructor Documentation 7

4.1.2.1 Andersen 7

4.1.3 Member Function Documentation 8

4.1.3.1 step 8

4.2 Atom Struct Reference 8

4.2.1 Detailed Description 8

4.3 FeneException Class Reference 9

4.3.1 Detailed Description 9

4.4 Integrator Class Reference 9

4.4.1 Detailed Description 10

4.5 Interaction Class Reference 10

4.5.1 Detailed Description 10

4.6 SljException Class Reference 10

4.6.1 Detailed Description 11

4.7 System Class Reference 11

4.7.1 Constructor & Destructor Documentation 13

4.7.1.1 System 13

4.7.2 Member Function Documentation 13

4.7.2.1 add_atom_type 13

4.7.2.2 add_atoms 13

4.7.2.3 add_atoms 14
4.7.2.4 add_bond 14
4.7.2.5 add_bond_type 14
4.7.2.6 add_ghost_atoms 14
4.7.2.7 atom_name 15
4.7.2.8 atom_type 15
4.7.2.9 bond_name 15
4.7.2.10 bond_type 15
4.7.2.11 clear_ghost_atoms 15
4.7.2.12 delete_atoms 15
4.7.2.13 gen_domain_info 16
4.8 Verlet Class Reference 16
4.8.1 Detailed Description 16
4.8.2 Constructor & Destructor Documentation 16
4.8.2.1 Verlet 16
4.8.3 Member Function Documentation 16
4.8.3.1 step 16

5 File Documentation 19
5.1 andersen.cpp File Reference 19
5.1.1 Detailed Description 19
5.1.2 Function Documentation 19
5.1.2.1 main 19
5.2 atom.cpp File Reference 19
5.2.1 Detailed Description 20
5.2.2 Function Documentation 20
5.2.2.1 create_MPI_ATOM 20
5.2.2.2 delete_MPI_atom 20
5.3 atom.h File Reference 20
5.3.1 Detailed Description 21
5.3.2 Function Documentation 21
5.3.2.1 create_MPI_ATOM 21
5.3.2.2 delete_MPI_atom 21
5.4 CBEMD.h File Reference 21
5.4.1 Detailed Description 21
5.5 common.h File Reference 22
5.5.1 Detailed Description 22
5.6 domain_decomp.cpp File Reference 22
5.6.1 Detailed Description 23
5.6.2 Function Documentation 23

5.6.2.1 communicate_skin_atoms 23
5.6.2.2 factorize 23
5.6.2.3 gen_goes_to 23
5.6.2.4 gen_send_lists 23
5.6.2.5 gen_send_table 24
5.6.2.6 gen_sets 24
5.6.2.7 gen_skin_cutoff 24
5.6.2.8 get_processor 24
5.6.2.9 get_processor 24
5.6.2.10 get_xyz_ids 24
5.6.2.11 init_domain_decomp 25
5.6.2.12 power 25
5.7 domain_decomp.h File Reference 25
5.7.1 Detailed Description 26
5.7.2 Function Documentation 26
5.7.2.1 communicate_skin_atoms 26
5.7.2.2 factorize 26
5.7.2.3 gen_goes_to 26
5.7.2.4 gen_send_lists 27
5.7.2.5 gen_send_table 27
5.7.2.6 gen_sets 27
5.7.2.7 get_processor 27
5.7.2.8 get_xyz_ids 28
5.7.2.9 init_domain_decomp 28
5.7.2.10 power 28
5.8 force_calc.cpp File Reference 28
5.8.1 Detailed Description 28
5.8.2 Function Documentation 29
5.8.2.1 force_calc 29
5.8.2.2 send_atoms 29
5.9 force_calc.h File Reference 29
5.9.1 Detailed Description 29
5.9.2 Function Documentation 29
5.9.2.1 force_calc 30
5.9.2.2 send_atoms 30
5.10 global.h File Reference 30
5.10.1 Detailed Description 31
5.11 initialize.cpp File Reference 31
5.11.1 Detailed Description 31
5.11.2 Function Documentation 31

5.11.2.1 abort_mpi . . . 31
5.11.2.2 end_mpi . . . 31
5.11.2.3 initialize_from_files . . . 32
5.11.2.4 start_mpi . . . 32
5.12 integrator.cpp File Reference . . . 32
5.12.1 Detailed Description . . . 32
5.12.2 Function Documentation . . . 32
5.12.2.1 run . . . 32
5.13 integrator.h File Reference . . . 33
5.13.1 Detailed Description . . . 33
5.13.2 Function Documentation . . . 33
5.13.2.1 run . . . 33
5.14 interaction.cpp File Reference . . . 33
5.14.1 Detailed Description . . . 34
5.14.2 Function Documentation . . . 34
5.14.2.1 fene . . . 34
5.14.2.2 harmonic . . . 34
5.14.2.3 slj . . . 35
5.15 interaction.h File Reference . . . 35
5.15.1 Detailed Description . . . 36
5.15.2 Function Documentation . . . 36
5.15.2.1 fene . . . 36
5.15.2.2 harmonic . . . 36
5.15.2.3 slj . . . 37
5.16 misc.cpp File Reference . . . 37
5.16.1 Detailed Description . . . 38
5.16.2 Function Documentation . . . 38
5.16.2.1 flag_error . . . 38
5.16.2.2 flag_notify . . . 38
5.16.2.3 mfopen . . . 38
5.16.2.4 min_image_dist2 . . . 39
5.16.2.5 min_image_dist2 . . . 39
5.16.2.6 pbc . . . 39
5.16.2.7 pbc . . . 39
5.16.2.8 unifRand . . . 39
5.17 misc.h File Reference . . . 40
5.17.1 Detailed Description . . . 40
5.17.2 Function Documentation . . . 40
5.17.2.1 flag_error . . . 40
5.17.2.2 flag_notify . . . 41

5.17.2.3 mfopen 41
5.17.2.4 min_image_dist2 41
5.17.2.5 min_image_dist2 41
5.17.2.6 pbc 41
5.17.2.7 pbc 42
5.17.2.8 unifRand 42
5.18 mpiatom.h File Reference 42
5.18.1 Detailed Description 42
5.19 read_interaction.cpp File Reference 42
5.19.1 Detailed Description 43
5.19.2 Function Documentation 43
5.19.2.1 get_fn 43
5.19.2.2 read_interactions 43
5.20 read_interaction.h File Reference 43
5.20.1 Detailed Description 44
5.20.2 Function Documentation 44
5.20.2.1 get_fn 44
5.20.2.2 read_interactions 44
5.21 read_xml.cpp File Reference 44
5.21.1 Detailed Description 44
5.21.2 Function Documentation 45
5.21.2.1 print_xml 45
5.21.2.2 read_xml 45
5.21.2.3 write_xyz 45
5.22 read_xml.h File Reference 45
5.22.1 Detailed Description 46
5.22.2 Function Documentation 46
5.22.2.1 print_xml 46
5.22.2.2 read_xml 46
5.22.2.3 write_xyz 46
5.23 system.cpp File Reference 47
5.23.1 Detailed Description 47
5.24 system.h File Reference 47
5.24.1 Detailed Description 47
5.25 verlet.cpp File Reference 47
5.25.1 Detailed Description 48
5.25.2 Function Documentation 48
5.25.2.1 main 48

Index **48**

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Atom 8
exception
    FeneException 9
    SljException 10
Integrator 9
    Andersen 7
    Verlet 16
Interaction 10
System 11

Chapter 2

Class Index

2.1 Class List

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

Andersen	NVT, Andersen thermostat Run (i.e. integrate) a system forward in time for a specified number of timesteps	7
Atom	Atom class is defined as a struct so as to be easy to pass with MPI	8
FeneException	Fene exception class is thrown if there is an error	9
Integrator	< Abstract base class for integrators	9
Interaction	This class stores how a pair of particles interacts	10
SljException	SLJ exception class is thrown if there is an error	10
System		11
Verlet	NVE, Verlet	16

Chapter 3

File Index

3.1 File List

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

andersen.cpp
Driver for MPI Version of CBEMD with Andersen Thermostat 19
atom.cpp
Functions for handling MD Atom Information 19
atom.h
MD Atom Information 20
CBEMD.h
Header grouping source headers into one simple header for the driver program(s) 21
common.h
Header that lumps all common header files into one 22
domain_decomp.cpp
Source code containing functions for performing domain decomposition 22
domain_decomp.h
Header file for domain decomposition 25
force_calc.cpp
Source code for force calculation 28
force_calc.h
Header for force_calc function 29
global.h
Global variables 30
initialize.cpp
Initialization routines for System object, as well as finalization of MPI 31
initialize.h . . . . . **??**
integrator.cpp
MD Integrator(s) Information 32
integrator.h
MD Integrator(s) Information 33
interaction.cpp
Source code for interaction functions 33
interaction.h
Header file for interaction information 35
misc.cpp
Source code for miscellaneous routines 37
misc.h
Header for Miscellaneous Routines 40
mpiatom.h 42
read_interaction.cpp
Source code to read in interaction parameters from a parameter file 42

read_interaction.h
Header file for reading interactions in 43
read_xml.cpp
I/O for XML file format 44
read_xml.h
I/O for XML 45
system.cpp
Source code for the System object 47
system.h
Header for MD System Information 47
verlet.cpp
Driver for MPI Version of CBEMD verlet 47

Chapter 4

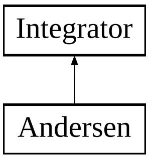
Class Documentation

4.1 Andersen Class Reference

NVT, Andersen thermostat Run (i.e. integrate) a system forward in time for a specified number of timesteps.

`#include <integrator.h>`

Inheritance diagram for Andersen:



Public Member Functions

- Andersen (double deltat, double temp, double nu)
- void **set_temp** (double temp)
- double **getTemp** () const
- double **getdt** ()
- int step (System ∗sys)
- double **nu** () const

Additional Inherited Members

4.1.1 Detailed Description

NVT, Andersen thermostat Run (i.e. integrate) a system forward in time for a specified number of timesteps.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Andersen::Andersen (double *deltat,* double *temp,* double *nu*)

Parameters

`in`	*deltat*	Incremental timestep
`in`	*temp*	Set point temperature

4.1.3 Member Function Documentation

4.1.3.1 int Andersen::step (System ∗ *sys*) `[virtual]`

Step forward one timestep with the Andersen thermostat. This reports the instantaneous temperature after each step.

Parameters

`in`	∗*sys*	Pointer to System to integrate.

Implements Integrator.

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

- integrator.h
- integrator.cpp

4.2 Atom Struct Reference

Atom class is defined as a struct so as to be easy to pass with MPI.

`#include <atom.h>`

Public Attributes

- double pos [NDIM]

  Cartesian coordinates.
- double prev_pos [NDIM]

  Cartesian coordinates for the previous position of the atom (needed for integrator)
- double vel [NDIM]

  Cartesian velocities (vx, vy, vz)
- double force [NDIM]

  Cartesian force, (fx, fy, fz)
- double mass

  Atomic mass (in reduced units)
- double diam

  Atomic diameter (in reduced units)
- int type

  Internally indexed type of this atom.
- int sys_index

  Global atom index, i.e. unique in the system.

4.2.1 Detailed Description

Atom class is defined as a struct so as to be easy to pass with MPI.

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

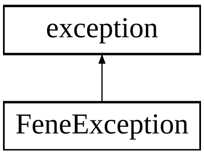
- atom.h

4.3 FeneException Class Reference

Fene exception class is thrown if there is an error.

`#include <interaction.h>`

Inheritance diagram for FeneException:



Public Member Functions

- **FeneException** (const int ind1, const int ind2, const double dist, const double r0)
- virtual const char $*$ **what** () const throw ()

Protected Attributes

- int **ind1_**
- int **ind2_**
- double **dist_**
- double **r0_**

4.3.1 Detailed Description

Fene exception class is thrown if there is an error.

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

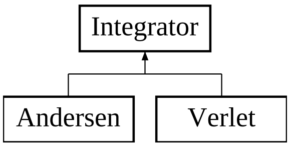
- interaction.h

4.4 Integrator Class Reference

< Abstract base class for integrators

`#include <integrator.h>`

Inheritance diagram for Integrator:



Public Member Functions

- void **set_dt** (const double dt)
- double **dt** () const
- void **set_temp** (double temp)

- double **getTemp** () const
- virtual int step (System ∗sys)=0

  Requires all subclasses to be able to execute a step.

Protected Attributes

- double **dt_**
- double **temp_**

4.4.1 Detailed Description

< Abstract base class for integrators

The step function should return SAFE_EXIT if successfully executed, integer error flag otherwise.

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

- integrator.h

4.5 Interaction Class Reference

This class stores how a pair of particles interacts.

`#include <interaction.h>`

Public Member Functions

- double force_energy (Atom ∗a1, Atom ∗a2, const vector< double > ∗box)

  Computes force (stored on atoms) and energy (returned)
- void set_force_energy (force_energy_ptr ife)

  Assign the potential calculator.
- void set_args (const vector< double > args)

  Assign energy and force arguments.
- force_energy_ptr check_force_energy_function () const

  Return the function for force and energy calculations.

4.5.1 Detailed Description

This class stores how a pair of particles interacts.

Returns energy and force as long as r < r_{cut}, else 0.

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

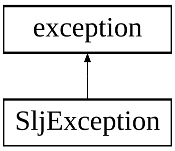
- interaction.h

4.6 SljException Class Reference

SLJ exception class is thrown if there is an error.

`#include <interaction.h>`

Inheritance diagram for SljException:



Public Member Functions

- **SljException** (const int ind1, const int ind2, const double dist, const double delta)
- virtual const char ∗ **what** () const throw ()

Protected Attributes

- int **ind1_**
- int **ind2_**
- double **dist_**
- double **delta_**

4.6.1 Detailed Description

SLJ exception class is thrown if there is an error.

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

- interaction.h

4.7 System Class Reference

Public Member Functions

- System ()
- void set_box (const vector< double > new_box)

  Set the global system box size.
- vector< double > box () const

  Report system size.
- void set_T (const double T)

  Set the system temperature.
- void set_P (const double P)

  Set the system pressure.
- double T () const

  Report the temperature of the system.
- double P () const

  Report the pressure of the system.
- int total_atoms () const

  Return the number of atoms currently in this system (processor) including current ghosts.
- int natoms () const

  Return the number of atoms this system (processor) is responsible for.
- int add_atom_type (const string atom_name)

  Index an atom name.
- int atom_type (const string atom_name)

Return the internal index associated with an atom name.
- string atom_name (const unsigned int index)

  Return the name associated with an index for atom type.
- int add_bond_type (const string bond_name)

  Index a bond name.
- int bond_type (const string bond_name)

  Return the internal index associated with a bond name.
- string bond_name (const unsigned int index)

  Return the name associated with an index for bond type.
- const pair< int, int > get_bond (const int nbond)

  Return a specific bonded pair indices.
- int get_bond_type (const int nbond)

  Return the internal index of a bond.
- int nbonds ()

  Return the number of bonds in the system.
- void add_bond (const int atom1, const int atom2, const int type)
- vector< int > add_atoms (const int natoms, Atom *new_atoms)

  Add atom(s) to the system.
- void add_ghost_atoms (const int natoms, Atom *new_atoms)

  Add ghost atom(s) to the system (does not update the number of atoms the processor is responsible for.
- vector< int > add_atoms (vector< Atom > *new_atoms)
- int delete_atoms (vector< int > indices)

  Pop atoms with local indices from local storage.
- Atom * get_atom (int index)

  Get pointer to atom by local index.
- Atom copy_atom (int index)

  Report a copy of an atom.
- void **set_rank** (int rank)
- int **rank** ()
- void **set_num_atoms** (int size)
- void clear_ghost_atoms ()
- int gen_domain_info ()
- void **set_max_rcut** (const double max_rcut)
- double **max_rcut** () const
- void set_total_KE (const double ke)

  Set the global kinetic energy record.
- void set_total_PE (const double pe)

  Set the global potential energy record.
- double **KE** () const
- double **U** () const

Public Attributes

- double proc_widths [NDIM]

  Width for domain decomposition.
- vector< int > final_proc_breakup

  Final domain decomposition.
- int **xyz_id** [NDIM]
- double **xyz_limits** [NDIM][2]
- int **send_table** [NNEIGHBORS]
- vector< vector< Atom > > **send_lists**
- int **send_list_size** [NNEIGHBORS]

- int **get_list_size** [NNEIGHBORS]
- vector< vector< Atom > > **get_lists**
- vector< vector< Interaction > > interact

  Interaction matrix between atoms indexed by global id's (symetric)
- vector< string > global_atom_types

  Keeps a record of every atom's type.

4.7.1 Constructor & Destructor Documentation

4.7.1.1 System::System ( )

Upon initialization, resize vectors as necessary. Set T < 0.

4.7.2 Member Function Documentation

4.7.2.1 int System::add_atom_type (const string *atom_name*)

Index an atom name.

Tries to add an atom type to the system, associating a user specified name with an internal index to reference this type in the future. This can return 3 different values:

> Returns 0 if, atom_name was new and was successfully indexed.

> Returns -1 if, atom_name was bad (i.e. empty string).

> Returns +1 if, atom_name already exists in the system and could not be indexed again.

Parameters

`in`	*atom_name*	User specified name to index

4.7.2.2 vector< int > System::add_atoms (const int *natoms,* Atom ∗ *new_atoms*)

Add atom(s) to the system.

Attempt to push an atom(s) into the system. This assigns the map automatically to link the atoms global index to the local storage location. This reallocates the internal vector that stores the atoms; if a memory error occurs during such reallocation, an error is given and the system exits.

Parameters

`in`	*natoms*	Length of the array of atoms to add to the system.
`in`	∗*new_atoms*	Pointer to an array of atoms the user has created elsewhere.

4.7.2.3 vector$<$ int $>$ System::add_atoms (vector$<$ Atom $>$ $*$ *new_atoms*)

Attempt to push an atom(s) into the system. This assigns the map automatically to link the atoms global index to the local storage location. This reallocates the internal vector that stores the atoms; if a memory error occurs during such reallocation, an error is given and the system exits.

Parameters

`in`	*natoms*	Length of the array of atoms to add to the system.
`in`	$*$*new_atoms*	Pointer to an array of atoms the user has created elsewhere.

4.7.2.4 void System::add_bond (const int *atom1,* const int *atom2,* const int *type*)

Adds a new bond and associated information to the System object.

Parameters

`in`	*atom1*	Global index of atom1 in the bond.
`in`	*atom1*	Global index of atom2 in the bond.
`in`	*type*	Internal index associated with this bond type.

4.7.2.5 int System::add_bond_type (const string *bond_name*)

Index a bond name.

Tries to add a bond type to the system, associating a user specified name with an internal index to reference this type in the future. This can return 3 different values:

> Returns 0 if, bond_name was new and was successfully indexed.
>
> Returns -1 if, bond_name was bad (i.e. empty string).
>
> Returns +1 if, bond_name already exists in the system and could not be indexed again.

Parameters

`in`	*bond_name*	User specified name to index

4.7.2.6 void System::add_ghost_atoms (const int *natoms,* Atom $*$ *new_atoms*)

Add ghost atom(s) to the system (does not update the number of atoms the processor is responsible for.

Attempt to push ghost atom(s) into the system. This assigns the map automatically to link the atoms global index to the local storage location. This reallocates the internal vector that stores the atoms; if a memory error occurs during such reallocation, an error is given and the system exits. Does not change num_atoms_ (the number of atoms a processor is responsible for)

Parameters

`in`	*natoms*	Length of the array of atoms to add to the system.
`in`	∗*new_atoms*	Pointer to an array of atoms the user has created elsewhere.

4.7.2.7 string System::atom_name (const unsigned int *index*)

Return the name associated with an index for atom type.

Returns the string "NULL" if failed, else user defined name of atom.

Parameters

`in`	*index*	Internal index to locate and return the name associated.

4.7.2.8 int System::atom_type (const string *name*)

Return the internal index associated with an atom name.

Returns the internal index associated with this atom name; returns -1 if not found.

Parameters

`in`	*name*	User defined name of atom type.

4.7.2.9 string System::bond_name (const unsigned int *index*)

Return the name associated with an index for bond type.

Returns the string "NULL" if failed, else user defined name of bond.

Parameters

`in`	*index*	Internal index to locate and return the name associated.

4.7.2.10 int System::bond_type (const string *name*)

Return the internal index associated with a bond name.

Returns the internal index associated with this bond name; returns -1 if not found.

Parameters

`in`	*name*	User defined name of bond type.

4.7.2.11 void System::clear_ghost_atoms ( )

Clears the atoms communicated from neighbouring domains from the list of atoms stored in the system leaving only the atoms the system is responsible for.

4.7.2.12 int System::delete_atoms (vector< int > *indices*)

Pop atoms with local indices from local storage.

Remove atoms from the system. Needs to sort indices because erase() operation reorders things; also, because of this it is fastest to pop from lowest to highest index. Returns the number of atoms deleted.

4.7.2.13 int System::gen_domain_info ( )

Generates the x,y,z ids for each processor and the absolute extents of the domain Returns 0 if successful, -1 if not.

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

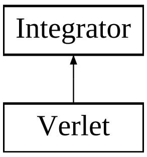
- system.h
- system.cpp

4.8 Verlet Class Reference

NVE, Verlet.

`#include <integrator.h>`

Inheritance diagram for Verlet:



Public Member Functions

- Verlet (double deltat)
- double **getTime** ()
- double **getdt** ()
- int step (System ∗sys)

Additional Inherited Members

4.8.1 Detailed Description

NVE, Verlet.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Verlet::Verlet (double *deltat*)

Parameters

`in`	*deltat*	Incremental timestep

4.8.3 Member Function Documentation

4.8.3.1 int Verlet::step (System ∗ *sys*) `[virtual]`

Parameters

`in,out`	**sys*	Pointer to System to make an integration step in.

Implements Integrator.

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

- integrator.h
- integrator.cpp

Chapter 5

File Documentation

5.1 andersen.cpp File Reference

Driver for MPI Version of CBEMD with Andersen Thermostat.

`#include "CBEMD.h"`

Functions

- int main (int argc, char ∗argv[])

5.1.1 Detailed Description

Driver for MPI Version of CBEMD with Andersen Thermostat.

Authors

{Nathan A. Mahynski, Carmeline Dsilva, Arun L. Prabhu, George Khoury, Frank Ricci, Jun Park}

5.1.2 Function Documentation

5.1.2.1 int main (int *argc*, char ∗ *argv[]*)

Parameters

∗*argv[]*	./andersen nsteps dt xml_file energy_file bond_file animation_file temperature nu

5.2 atom.cpp File Reference

Functions for handling MD Atom Information.

`#include "atom.h"`

Functions

- void create_MPI_ATOM ()

  Creates the MPI_Atom class so it can be passed with MPI.

- void delete_MPI_atom ()

  Free the MPI type at the end of the program.

5.2.1 Detailed Description

Functions for handling MD Atom Information.

Author

{Nathan A. Mahynski, Carmeline Dsilva}

5.2.2 Function Documentation

5.2.2.1 void create_MPI_ATOM ( )

Creates the MPI_Atom class so it can be passed with MPI.

This function creates and commits to memory the MPI_ATOM derived data type. (see example of use at `https://computing.llnl.gov/tutorials/mpi/#Derived_Data_Types`)

See Also

initialize, MPI_ATOM

5.2.2.2 void delete_MPI_atom ( )

Free the MPI type at the end of the program.

This function utilizes the complementary MPI_Type_free routine to mark the MPI_ATOM type for deallocation at the end of the program.

See Also

finalize

5.3 atom.h File Reference

MD Atom Information.

```
#include "mpi.h"
#include "global.h"
```

Classes

- struct Atom

  Atom class is defined as a struct so as to be easy to pass with MPI.

Functions

- void create_MPI_ATOM ()

  Creates the MPI_Atom class so it can be passed with MPI.
- void delete_MPI_atom ()

  Free the MPI type at the end of the program.

5.3.1 Detailed Description

MD Atom Information.

Author

Nathan A. Mahynski

5.3.2 Function Documentation

5.3.2.1 void create_MPI_ATOM ( )

Creates the MPI_Atom class so it can be passed with MPI.

This function creates and commits to memory the MPI_ATOM derived data type. (see example of use at `https://computing.llnl.gov/tutorials/mpi/#Derived_Data_Types`)

See Also

initialize, MPI_ATOM

5.3.2.2 void delete_MPI_atom ( )

Free the MPI type at the end of the program.

This function utilizes the complementary MPI_Type_free routine to mark the MPI_ATOM type for deallocation at the end of the program.

See Also

finalize

5.4 CBEMD.h File Reference

Header grouping source headers into one simple header for the driver program(s).

```
#include "atom.h"
#include "system.h"
#include "integrator.h"
#include "misc.h"
#include "global.h"
#include "read_xml.h"
#include "interaction.h"
#include "mpiatom.h"
#include "initialize.h"
#include "mpi.h"
#include <limits>
```

5.4.1 Detailed Description

Header grouping source headers into one simple header for the driver program(s).

Author

{Carmeline Dsilva}

5.5 common.h File Reference

Header that lumps all common header files into one.

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <string>
```

5.5.1 Detailed Description

Header that lumps all common header files into one.

5.6 domain_decomp.cpp File Reference

Source code containing functions for performing domain decomposition.

```
#include "domain_decomp.h"
```

Functions

- void gen_sets (const vector< int > &factors, const double box[ ], const int level, double &final_diff, vector< int > &final_breakup, int add)

  Given the box size and factors of nprocs, checks which combination generates the most cubic domains.
- vector< int > factorize (const int nprocs)

  Given a number returns the prime factors (does not count 1 as a prime factor)
- int init_domain_decomp (const vector< double > box, const int nprocs, double widths[ ], vector< int > &final_breakup)

  Decomposes the box into domains for each processor to handle.
- int get_processor (const vector< double > pos, const System ∗sys)

  Given the co-ordinates of a point, determines within which domain the point lies.
- int get_processor (const int x_id, const int y_id, const int z_id, const vector< int > &final_breakup)
- int get_xyz_ids (const int domain_id, const vector< int > &final_breakup, int xyz_id[ ])

  Given a domain_id specifies the x, y, z ids of the domain.
- int gen_send_lists (System ∗sys)

  Generates the lists of molecules that need to be passed to other processors.
- int gen_send_table (System ∗sys)

  Given the rank, generates the list of its neighbours.
- double gen_skin_cutoff ()
- void gen_goes_to (const vector< int > &is_near_border, vector< int > &goes_to, const int ndims)

  Generates the list of neighbours a particle should be sent to based on the borders its near.
- int power (int base, int exponent)

  Computes the exponentiation of an integer by an integral power.
- int communicate_skin_atoms (System ∗sys)

  Communicates the atoms in the skin regions of the processors to the appropriate neighbours.

5.6.1 Detailed Description

Source code containing functions for performing domain decomposition.

Author

Arun L. Prabhu

5.6.2 Function Documentation

5.6.2.1 int communicate_skin_atoms (**System** ∗ *sys*)

Communicates the atoms in the skin regions of the processors to the appropriate neighbours.

Communicates the atoms in the skin regions of the processors to the appropriate neighbours

Parameters

sys	[in,out] System to be evaluated

5.6.2.2 vector<int> factorize (const int *nprocs*)

Given a number returns the prime factors (does not count 1 as a prime factor)

Given a number returns the prime factors of that number. This function does not count 1 as a prime factor.

Parameters

`in`	*nprocs*	number of processors

5.6.2.3 void gen_goes_to (const vector< int > & *is_near_border,* vector< int > & *goes_to,* const int *ndims*)

Generates the list of neighbours a particle should be sent to based on the borders its near.

Generates the list of neighbours a particle should be sent to based on the borders its near

Parameters

`in`	*is_near_border*	WHAT DOES IT DO
	goes_to	[in] WHAT DOES IT DO
`in`	*ndims*	WHAT DOES IT DO

5.6.2.4 int gen_send_lists (**System** ∗ *sys*)

Generates the lists of molecules that need to be passed to other processors.

Generates the lists of molecules that need to be passed to other processors

Parameters

`in,out`	*sys*	WHAT DOES IT DO
`in`	*rank*	rank of processor
`in`	*skin_cutoff*	WHAT DOES IT DO

5.6.2.5 int gen_send_table (System * *sys*)

Given the rank, generates the list of its neighbours.

Given the rank, generates the list of its neighbours Assumes 3D system, for different number of dimensions need to make this a recursive function

Parameters

`in`	*sys*	WHAT DOES IT DO

5.6.2.6 void gen_sets (const vector< int > & *factors,* const double *box[ ],* const int *level,* double & *final_diff,* vector< int > & *final_breakup,* int *add*)

Given the box size and factors of nprocs, checks which combination generates the most cubic domains.

Given the box size and factors of nprocs, this recursive function checks which combination generates the most cubic domains

Parameters

`in`	*factors*	WHAT DOES IT DO
`in`	*box*	WHAT DOES IT DO
`in`	*level*	WHAT DOES IT DO
`in`	*final_breakup*	WHAT DOES IT DO

5.6.2.7 double gen_skin_cutoff ( )

Generates the skin cutoff distance based on the largest cutoff in the system

5.6.2.8 int get_processor (const vector< double > *pos,* const System * *sys*)

Given the co-ordinates of a point, determines within which domain the point lies.

Given the coordinates of a point, determines within which domain the point lies. This function is overloaded.

Parameters

`in`	*pos*	the vector containing the coordinates of a point
`in`	*sys*	system passed to be able to utilize final domain decomposition

5.6.2.9 int get_processor (const int *x_id,* const int *y_id,* const int *z_id,* const vector< int > & *final_breakup*)

Given the coordinates of a point, determines within which domain the point lies. This function is overloaded.

Parameters

`in`	*x_id*	WHAT DOES IT DO
`in`	*y_id*	WHAT DOES IT DO
`in`	*z_id*	WHAT DOES IT DO
`in`	*fina_breakup*	WHAT DOES IT DO

5.6.2.10 int get_xyz_ids (const int *domain_id,* const vector< int > & *final_breakup,* int *xyz_id[ ]*)

Given a domain_id specifies the x, y, z ids of the domain.

Given a domain_id specifies the x, y, z ids of the domain

Parameters

`in`	*domain_id*	WHAT DOES IT DO
`in`	*final_breakup*	WHAT DOES IT DO
`in`	*xyz_id*	WHAT DOES IT DO

5.6.2.11 int init_domain_decomp (const vector< double > *box*, const int *nprocs*, double *widths[ ]*, vector< int > & *final_breakup*)

Decomposes the box into domains for each processor to handle.

Decomposes the box into domains for each processor to handle

Parameters

`in`	*box*	WHAT DOES IT DO
`in`	*nprocs*	number of processors
`in`	*widths*	WHAT DOES IT DO
`out`	*final_breakup*	WHAT DOES IT DO

5.6.2.12 int power (int *base*, int *exponent*)

Computes the exponentiation of an integer by an integral power.

Computes the exponentiation of an integer by an integral power

Parameters

`in`	*base*	the base value to be raised to a power
`in`	*exponent*	the exponent of the power base is raised to

5.7 domain_decomp.h File Reference

Header file for domain decomposition.

```
#include "common.h"
#include "system.h"
#include "mpi.h"
```

Functions

- void gen_sets (const vector< int > &factors, const double box[ ], const int level, double &final_diff, vector< int > &final_breakup, int add)

  Given the box size and factors of nprocs, checks which combination generates the most cubic domains.
- vector< int > factorize (const int nprocs)

  Given a number returns the prime factors (does not count 1 as a prime factor)
- int init_domain_decomp (const vector< double > box, const int nprocs, double widths[ ], vector< int > &final_breakup)

  Decomposes the box into domains for each processor to handle.
- int get_processor (const vector< double > pos, const System ∗sys)

  Given the co-ordinates of a point, determines within which domain the point lies.

- int get_processor_id (const int x_id, const int y_id, const int z_id, const vector< int > &final_breakup)

  Given the x, y, z ids of a domain, determines the domain id (useful for locating neighbouring domains)
- int get_xyz_ids (const int domain_id, const vector< int > &final_breakup, int xyz_id[ ])

  Given a domain_id specifies the x, y, z ids of the domain.
- int gen_send_lists (System ∗sys)

  Generates the lists of molecules that need to be passed to other processors.
- int gen_send_table (System ∗sys)

  Given the rank, generates the list of its neighbours.
- double gen_skin_cutoff (...)

  Generates the skin cutoff distance based on the largest cutoff in the system.
- void gen_goes_to (const vector< int > &is_near_border, vector< int > &goes_to, const int ndims)

  Generates the list of neighbours a particle should be sent to based on the borders its near.
- int power (int base, int exponent)

  Computes the exponentiation of an integer by an integral power.
- int communicate_skin_atoms (System ∗sys)

  Communicates the atoms in the skin regions of the processors to the appropriate neighbours.

5.7.1 Detailed Description

Header file for domain decomposition.

Author

Arun L. Prabhu

5.7.2 Function Documentation

5.7.2.1 int communicate_skin_atoms (System ∗ *sys*)

Communicates the atoms in the skin regions of the processors to the appropriate neighbours.

Communicates the atoms in the skin regions of the processors to the appropriate neighbours

Parameters

sys	[in,out] System to be evaluated

5.7.2.2 vector<int> factorize (const int *nprocs*)

Given a number returns the prime factors (does not count 1 as a prime factor)

Given a number returns the prime factors of that number. This function does not count 1 as a prime factor.

Parameters

`in`	*nprocs*	number of processors

5.7.2.3 void gen_goes_to (const vector< int > & *is_near_border,* vector< int > & *goes_to,* const int *ndims*)

Generates the list of neighbours a particle should be sent to based on the borders its near.

Generates the list of neighbours a particle should be sent to based on the borders its near

Parameters

`in`	*is_near_border*	WHAT DOES IT DO
	goes_to	[in] WHAT DOES IT DO
`in`	*ndims*	WHAT DOES IT DO

5.7.2.4 int gen_send_lists (**System** ∗ *sys*)

Generates the lists of molecules that need to be passed to other processors.

Generates the lists of molecules that need to be passed to other processors

Parameters

`in,out`	*sys*	WHAT DOES IT DO
`in`	*rank*	rank of processor
`in`	*skin_cutoff*	WHAT DOES IT DO

5.7.2.5 int gen_send_table (**System** ∗ *sys*)

Given the rank, generates the list of its neighbours.

Given the rank, generates the list of its neighbours Assumes 3D system, for different number of dimensions need to make this a recursive function

Parameters

`in`	*sys*	WHAT DOES IT DO

5.7.2.6 void gen_sets (const vector< int > & *factors,* const double *box[ ],* const int *level,* double & *final_diff,* vector< int > & *final_breakup,* int *add*)

Given the box size and factors of nprocs, checks which combination generates the most cubic domains.

Given the box size and factors of nprocs, this recursive function checks which combination generates the most cubic domains

Parameters

`in`	*factors*	WHAT DOES IT DO
`in`	*box*	WHAT DOES IT DO
`in`	*level*	WHAT DOES IT DO
`in`	*final_breakup*	WHAT DOES IT DO

5.7.2.7 int get_processor (const vector< double > *pos,* const **System** ∗ *sys*)

Given the co-ordinates of a point, determines within which domain the point lies.

Given the coordinates of a point, determines within which domain the point lies. This function is overloaded.

Parameters

`in`	*pos*	the vector containing the coordinates of a point
`in`	*sys*	system passed to be able to utilize final domain decomposition

5.7.2.8 int get_xyz_ids (const int *domain_id*, const vector< int > & *final_breakup*, int *xyz_id[]*)

Given a domain_id specifies the x, y, z ids of the domain.

Given a domain_id specifies the x, y, z ids of the domain

Parameters

`in`	*domain_id*	WHAT DOES IT DO
`in`	*final_breakup*	WHAT DOES IT DO
`in`	*xyz_id*	WHAT DOES IT DO

5.7.2.9 int init_domain_decomp (const vector< double > *box*, const int *nprocs*, double *widths[]*, vector< int > & *final_breakup*)

Decomposes the box into domains for each processor to handle.

Decomposes the box into domains for each processor to handle

Parameters

`in`	*box*	WHAT DOES IT DO
`in`	*nprocs*	number of processors
`in`	*widths*	WHAT DOES IT DO
`out`	*final_breakup*	WHAT DOES IT DO

5.7.2.10 int power (int *base*, int *exponent*)

Computes the exponentiation of an integer by an integral power.

Computes the exponentiation of an integer by an integral power

Parameters

`in`	*base*	the base value to be raised to a power
`in`	*exponent*	the exponent of the power base is raised to

5.8 force_calc.cpp File Reference

Source code for force calculation.

`#include "force_calc.h"`

Functions

- int send_atoms (System ∗sys)

  Move atoms between processors (domains)
- int force_calc (System ∗sys)

  Calculates the forces between the particles in the system.

5.8.1 Detailed Description

Source code for force calculation.

Authors

{Frank Ricci, Jun Park, Nathan Mahynski, Carmeline Dsilva, Arun Prabhu, George Khoury}

5.8.2 Function Documentation

5.8.2.1 int force_calc (**System** ∗ *sys*)

Calculates the forces between the particles in the system.

Returns SAFE_EXIT if successful, else returns an error flag.

Parameters

`in`	∗*sys*	Pointer to system for which to evaluate the forces

5.8.2.2 int send_atoms (**System** ∗ *sys*)

Move atoms between processors (domains)

This function sends the atoms that have left the domain of the processor to the relevant neighboring processor. If the atom has moved more than one domain width, it returns an error flag, else returns SAFE_EXIT for success.

Parameters

∗*sys*	[in] Pointer to system for which to move the atoms from

5.9 force_calc.h File Reference

Header for force_calc function.

```
#include "system.h"
#include "atom.h"
#include "interaction.h"
```

Functions

- int force_calc (System ∗sys)

  Calculates the forces between the particles in the system.
- int send_atoms (System ∗sys)

  Move atoms between processors (domains)

5.9.1 Detailed Description

Header for force_calc function.

Author

{Frank Ricci, Jun Park}

5.9.2 Function Documentation

5.9.2.1 int force_calc (**System** ∗ *sys*)

Calculates the forces between the particles in the system.

Returns SAFE_EXIT if successful, else returns an error flag.

Parameters

`in`	∗*sys*	Pointer to system for which to evaluate the forces

5.9.2.2 int send_atoms (**System** ∗ *sys*)

Move atoms between processors (domains)

This function sends the atoms that have left the domain of the processor to the relevant neighboring processor. If the atom has moved more than one domain width, it returns an error flag, else returns SAFE_EXIT for success.

Parameters

∗*sys*	[in] Pointer to system for which to move the atoms from

5.10 global.h File Reference

Global variables.

```
#include <exception>
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <string>
```

Enumerations

- enum NAME_SIZES { **ATOM_NAME_LENGTH** = 100, **BOND_NAME_LENGTH** = 200, **MYERR_FLAG_SIZE** = 1000 }

  Set maximum size for user-defined names of things.
- enum RETURN_FLAGS {
  SAFE_EXIT = 0, **BAD_MEM** = 1, **ILLEGAL_VALUE** = 2, **MPI_FAIL** = 3,
  INTEGRATE_FAIL = 4, **FILE_ERROR** = 5, **BAD_EXIT** = 6 }

  Set error flags returned if a failure condition is met.

Variables

- const double WCA_CUTOFF = pow(2.0, 1/6.0)

  The Weeks-Chandler-Andersen cutoff length is a fixed number that is used in numerous places commonly in MD so it is precomputed here.
- MPI_Datatype MPI_ATOM

  MPI_ATOM is made visible to all routines.
- const int NDIM = 3

  Number of dimensions in the system (3D), but future work may include 2D as well. This leaves the door open for this.
- const int NNEIGHBORS = 26

Nearest neighbors for 3D Domain decomposition.

- const int PARALLELDIM = 0

  dimension along which to do the domain decomposition

5.10.1 Detailed Description

Global variables.

Authors

{Nathan A. Mahynski, George Khoury}

5.11 initialize.cpp File Reference

Initialization routines for System object, as well as finalization of MPI.

`#include "initialize.h"`

Functions

- int initialize_from_files (const string xml_filename, const string energy_filename, System *sys)

  Parse an XML and energy file to obtain atom and interaction information.
- int start_mpi (int argc, char *argv[])

  Call MPI_Init and start the MPI ensuring it began successfully.
- int end_mpi ()

  Finalize MPI when program finishes successfully.
- int abort_mpi ()

  Abort MPI in case of failure.

5.11.1 Detailed Description

Initialization routines for System object, as well as finalization of MPI.

Author

Nathan A. Mahynski

5.11.2 Function Documentation

5.11.2.1 int abort_mpi ( )

Abort MPI in case of failure.

Abort the MPI as a result of a failure, freeing MPI_ATOM as necessary.

5.11.2.2 int end_mpi ( )

Finalize MPI when program finishes successfully.

Finalize MPI and clean up derived data types.

5.11.2.3 int initialize_from_files (const string *xml_filename*, const string *energy_filename*, System ∗ *sys*)

Parse an XML and energy file to obtain atom and interaction information.

Parse an XML and energy file to obtain atom and interaction information. Returns 0 if successful, non-zero if failure. Operates in a "cascade" between ranks so that each processor (if MPI is used) opens and initializes from the input file in order.

Parameters

`in`	*xml_filename*	Name of coordinate file to open and read.
`in`	*energy_filename*	Name of file containing bonds, pair potential parameters, etc.
`in,out`	∗*sys*	Pointer to System object to store this information in.

5.11.2.4 int start_mpi (int *argc*, char ∗ *argv[]*)

Call MPI_Init and start the MPI ensuring it began successfully.

Call MPI_Init and start the MPI ensuring it began successfully. Returns SAFE_EXIT if successful.

Parameters

`in`	*argc*	Number of arguments in ∗argv[].
`in`	∗*argv[]*	Array of character arguments.

5.12 integrator.cpp File Reference

MD Integrator(s) Information.

`#include "integrator.h"`

Functions

- int run (System ∗sys, Integrator ∗integrator, const int timesteps, const string outfile)

5.12.1 Detailed Description

MD Integrator(s) Information.

Authors

{George Khoury, Carmeline Dsilva, Nathan A. Mahynski}

5.12.2 Function Documentation

5.12.2.1 int run (System ∗ *sys*, Integrator ∗ *integrator*, const int *timesteps*, const string *outfile*)

Function returns SAFE_EXIT if successful, error flag otherwise.

Parameters

`in,out`	∗*sys*	Pointer to System to integrate
`in`	∗*integrator*	Pointer to Integrator to use
`in`	*timesteps*	Number of timesteps to integrate over
`in`	*outfile*	Name of file to print animation to

5.13 integrator.h File Reference

MD Integrator(s) Information.

```
#include "system.h"
#include "misc.h"
#include "mpi.h"
#include "atom.h"
#include "global.h"
#include "force_calc.h"
#include "read_xml.h"
#include <boost/tr1/random.hpp>
```

Classes

- class Integrator

  < Abstract base class for integrators
- class Verlet

  NVE, Verlet.
- class Andersen

  NVT, Andersen thermostat Run (i.e. integrate) a system forward in time for a specified number of timesteps.

Functions

- int run (System ∗sys, Integrator ∗integrator, const int timesteps, const string outfile)

5.13.1 Detailed Description

MD Integrator(s) Information.

Author

{Nathan A. Mahynski, George Khoury}

5.13.2 Function Documentation

5.13.2.1 int run (**System** ∗ *sys,* **Integrator** ∗ *integrator,* const int *timesteps,* const string *outfile*)

Function returns SAFE_EXIT if successful, error flag otherwise.

Parameters

`in,out`	*∗sys*	Pointer to System to integrate
`in`	*∗integrator*	Pointer to Integrator to use
`in`	*timesteps*	Number of timesteps to integrate over
`in`	*outfile*	Name of file to print animation to

5.14 interaction.cpp File Reference

Source code for interaction functions.

```
#include "interaction.h"
```

Functions

- double slj (Atom ∗atom1, Atom ∗atom2, const vector< double > ∗box, const vector< double > ∗args)

  Shifted Lennard-Jones Force.
- double harmonic (Atom ∗a1, Atom ∗a2, const vector< double > ∗box, const vector< double > ∗args)

  Computes force and energy of a Harmonic bond.
- double fene (Atom ∗a1, Atom ∗a2, const vector< double > ∗box, const vector< double > ∗args)

  Computes force and energy of a Fene bond.

5.14.1 Detailed Description

Source code for interaction functions.

Authors

{Nathan A. Mahynski, George Khoury}

5.14.2 Function Documentation

5.14.2.1 double fene (**Atom** ∗ *a1,* **Atom** ∗ *a2,* const vector< double > ∗ *box,* const vector< double > ∗ *args*)

Computes force and energy of a Fene bond.

Finitely Extensible Non-linear Elastic Bond (FENE) The Fene bond potential is given by:

$$U(r) = -\frac{1}{2}kr_0^2 ln\left(1-\left(\frac{r-\Delta}{r_0}\right)^2\right) + U_{WCA}$$

Where the short range repulsion is provided by the WCA potential:

$$\begin{aligned} U_{WCA} &= 4\varepsilon\left(\left(\frac{\sigma}{r-\Delta}\right)^{12} - \left(\frac{\sigma}{r-\Delta}\right)^{6}\right) + \varepsilon & \text{r < } 2^{1/6}\sigma+\Delta \\ &= 0 & \text{r-}\Delta \geq 2^{1/6}\sigma \end{aligned}$$

is usually set such that $ = (d_i+d_j)/2-1$ where d_i is the diameter of species i, but the user may decide on other parameters.

Parameters

`in,out`	*∗atom1*	Pointer to first atom
`in,out`	*∗atom2*	Pointer to second atom
`in`	*∗box*	Pointer to vector of box size
`in`	*∗args*	Vector of arguments <epsilon, sigma, delta, k, r0>

5.14.2.2 double harmonic (**Atom** ∗ *a1,* **Atom** ∗ *a2,* const vector< double > ∗ *box,* const vector< double > ∗ *args*)

Computes force and energy of a Harmonic bond.

Harmonic Bond The Harmonic bond potential is given by:

$$U(r) = \frac{1}{2}k(r-r_0)^2$$

Parameters

`in,out`	*$*$atom1*	Pointer to first atom
`in,out`	*$*$atom2*	Pointer to second atom
`in`	*$*$box*	Pointer to vector of box size
`in`	*$*$args*	Vector of arguments <k, r0>

5.14.2.3 double slj (**Atom** $*$ *atom1*, **Atom** $*$ *atom2*, const vector< double > $*$ *box*, const vector< double > $*$ *args*)

Shifted Lennard-Jones Force.

Computes force and energy of Shifted Lennard-Jones interaction.

This is the same as standard LJ if = 0. This is generally useful for systems with large size asymmetries. The energy U(r) is returned:

$$\begin{aligned} U(r) &= 4\varepsilon\left(\left(\frac{\sigma}{r-\Delta}\right)^{12}-\left(\frac{\sigma}{r-\Delta}\right)^{6}\right)+U_{shift} && \text{r - }\Delta < r_{cut} \\ &= 0 && \text{r - }\Delta \geq r_{cut} \end{aligned}$$

Forces are added to atoms:

$$\begin{aligned} F_i &= -\frac{U}{r}\frac{r}{x_i} = -\frac{U}{r}\frac{x_i}{r} && \text{r - }\Delta < r_{cut} \\ &= 0 && \text{r - }\Delta \geq r_{cut} \end{aligned}$$

Parameters

`in,out`	*$*$atom1*	Pointer to first atom
`in,out`	*$*$atom2*	Pointer to second atom
`in`	*$*$box*	Pointer to vector of box size
`in`	*$*$args*	Pointer to vector of arguments <epsilon, sigma, delta, U_{shift}, rcut^2>

5.15 interaction.h File Reference

Header file for interaction information.

```
#include "atom.h"
#include "misc.h"
#include "global.h"
```

Classes

- class Interaction

  This class stores how a pair of particles interacts.
- class FeneException

  Fene exception class is thrown if there is an error.
- class SljException

  SLJ exception class is thrown if there is an error.

Typedefs

- typedef double($*$ **force_energy_ptr**)(Atom $*$a1, Atom $*$a2, const vector< double > $*$box, const vector< double > $*$args)

Functions

- double slj (Atom ∗a1, Atom ∗a2, const vector< double > ∗box, const vector< double > ∗args)

  Computes force and energy of Shifted Lennard-Jones interaction.
- double fene (Atom ∗a1, Atom ∗a2, const vector< double > ∗box, const vector< double > ∗args)

  Computes force and energy of a Fene bond.
- double harmonic (Atom ∗a1, Atom ∗a2, const vector< double > ∗box, const vector< double > ∗args)

  Computes force and energy of a Harmonic bond.

5.15.1 Detailed Description

Header file for interaction information.

Author

Nathan A. Mahynski

5.15.2 Function Documentation

5.15.2.1 double fene (**Atom** ∗ *a1,* **Atom** ∗ *a2,* const vector< double > ∗ *box,* const vector< double > ∗ *args*)

Computes force and energy of a Fene bond.

Finitely Extensible Non-linear Elastic Bond (FENE) The Fene bond potential is given by:

$$U(r) = -\frac{1}{2}kr_0^2 ln\left(1-\left(\frac{r-\Delta}{r_0}\right)^2\right)+U_{WCA}$$

Where the short range repulsion is provided by the WCA potential:

$$\begin{array}{rcll} U_{WCA} & = & 4\varepsilon\left(\left(\frac{\sigma}{r-\Delta}\right)^{12}-\left(\frac{\sigma}{r-\Delta}\right)^{6}\right)+\varepsilon & \text{r < } 2^{1/6}\sigma+\Delta \\ & = & 0 & \text{r-}\Delta \geq 2^{1/6}\sigma \end{array}$$

is usually set such that $ = (d_i+d_j)/2-1$ where d_i is the diameter of species i, but the user may decide on other parameters.

Parameters

`in,out`	∗*atom1*	Pointer to first atom
`in,out`	∗*atom2*	Pointer to second atom
`in`	∗*box*	Pointer to vector of box size
`in`	∗*args*	Vector of arguments <epsilon, sigma, delta, k, r0>

5.15.2.2 double harmonic (**Atom** ∗ *a1,* **Atom** ∗ *a2,* const vector< double > ∗ *box,* const vector< double > ∗ *args*)

Computes force and energy of a Harmonic bond.

Harmonic Bond The Harmonic bond potential is given by:

$$U(r) = \frac{1}{2}k(r-r_0)^2$$

Parameters

`in,out`	∗*atom1*	Pointer to first atom
`in,out`	∗*atom2*	Pointer to second atom
`in`	∗*box*	Pointer to vector of box size
`in`	∗*args*	Vector of arguments <k, r0>

5.15.2.3 double slj (Atom ∗ *atom1*, Atom ∗ *atom2*, const vector< double > ∗ *box*, const vector< double > ∗ *args*)

Computes force and energy of Shifted Lennard-Jones interaction.

Computes force and energy of Shifted Lennard-Jones interaction.

This is the same as standard LJ if = 0. This is generally useful for systems with large size asymmetries. The energy U(r) is returned:

$$\begin{aligned} U(r) &= 4\varepsilon\left(\left(\frac{\sigma}{r-\Delta}\right)^{12} - \left(\frac{\sigma}{r-\Delta}\right)^{6}\right) + U_{shift} && \text{r - }\Delta < r_{cut} \\ &= 0 && \text{r - }\Delta \geq r_{cut} \end{aligned}$$

Forces are added to atoms:

$$\begin{aligned} F_i &= -\frac{U}{r}\frac{r}{x_i} = -\frac{U}{r}\frac{x_i}{r} && \text{r - }\Delta < r_{cut} \\ &= 0 && \text{r - }\Delta \geq r_{cut} \end{aligned}$$

Parameters

`in,out`	∗*atom1*	Pointer to first atom
`in,out`	∗*atom2*	Pointer to second atom
`in`	∗*box*	Pointer to vector of box size
`in`	∗*args*	Pointer to vector of arguments <epsilon, sigma, delta, U_{shift}, rcut^2>

5.16 misc.cpp File Reference

Source code for miscellaneous routines.

`#include "misc.h"`

Functions

- void flag_error (const char ∗msg, const char ∗file, const int line)

  Report an error message.
- void flag_notify (const char ∗msg, const char ∗file, const int line)

  Report a notification.
- FILE ∗ mfopen (const char ∗filename, const char ∗opt)

  Safely open a file.
- vector< double > pbc (const vector< double > coords, const vector< double > box)

  Returns the equivalent cartesian coordinates back in the simulation box assuming periodic boundaries.
- vector< double > pbc (const double ∗coords, const vector< double > box)

  Returns the equivalent cartesian coordinates back in the simulation box assuming periodic boundaries.
- double min_image_dist2 (const vector< double > coords1, const vector< double > coords2, const vector< double > box)

Return the square of the minimum image distance between 2 coordinate vectors.

- double min_image_dist2 (const Atom ∗a1, const Atom ∗a2, const vector< double > ∗box, double ∗xyz)

  Returns the square of the minimum image distance between 2 atoms, also returns the minimum image distance vector xyz that points from atom1 to atom2.
- double unifRand ()

  Generate a random number between 0 and 1, returns a uniform number in [0,1].
- double unifRand (double a, double b)

  Generate a random number in a real interval.

5.16.1 Detailed Description

Source code for miscellaneous routines.

Author

Nathan A. Mahynski

5.16.2 Function Documentation

5.16.2.1 void flag_error (const char ∗ *msg*, const char ∗ *file*, const int *line*)

Report an error message.

Error messages are sent to stderr.

Parameters

in	∗*msg*	Character string to print out.
in	∗*file*	**FILE** this function is called from.
in	*line*	**LINE** this function is called from.

5.16.2.2 void flag_notify (const char ∗ *msg*, const char ∗ *file*, const int *line*)

Report a notification.

Notification messages are sent to stderr.

Parameters

in	∗*msg*	Character string to print out.
in	∗*file*	**FILE** this function is called from.
in	*line*	**LINE** this function is called from.

5.16.2.3 FILE∗ mfopen (const char ∗ *filename*, const char ∗ *opt*)

Safely open a file.

Tries to open a file, if it fails it returns a NULL pointer and alerts the user with an error message. If it succeeds it returns the file pointer.

Parameters

in	∗*filename*	Character name of file to open.
in	∗*opt*	File option ("r","w","rw+",etc.).

5.16.2.4 double min_image_dist2 (const vector< double > *coords1*, const vector< double > *coords2*, const vector< double > *box*)

Return the square of the minimum image distance between 2 coordinate vectors.

Parameters

`in`	*coords1*	Vector of cartesian coordinates of one atom
`in`	*coords2*	Vector of cartesian coordinates of the other atom
`in`	*box*	Vector of cartesian coordinates of the box

5.16.2.5 double min_image_dist2 (const Atom ∗ *a1*, const Atom ∗ *a2*, const vector< double > ∗ *box*, double ∗ *xyz*)

Returns the square of the minimum image distance between 2 atoms, also returns the minimum image distance vector xyz that points from atom1 to atom2.

Parameters

`in`	∗*a1*	Pointer to one atom
`in`	∗*a2*	Pointer to the other atom
`in`	*box*	Vector of cartesian coordinates of the box
`in,out`	∗*xyz*	Array of xyz displacements to be returned to the user (length 3)

5.16.2.6 vector<double> pbc (const vector< double > *coords*, const vector< double > *box*)

Returns the equivalent cartesian coordinates back in the simulation box assuming periodic boundaries.

If this routine fails, it returns an empty vector (size = 0).

Parameters

`in`	*coords*	Vector of cartesian coordinates.
`in`	*box*	Vector of box dimensions (L_x, L_y, L_z).

5.16.2.7 vector<double> pbc (const double ∗ *coords*, const vector< double > *box*)

Returns the equivalent cartesian coordinates back in the simulation box assuming periodic boundaries.

If this routine fails, it returns an empty vector (size = 0).

Parameters

`in`	∗*coords*	Array of cartesian coordinates.
`in`	*box*	Vector of box dimensions (L_x, L_y, L_z).

5.16.2.8 double unifRand (double *a*, double *b*)

Generate a random number in a real interval.

Parameters

a	Lower end point of the interval
b	Upper end of the interval

5.17 misc.h File Reference

Header for Miscellaneous Routines.

```
#include <map>
#include <assert.h>
#include "atom.h"
#include "global.h"
```

Functions

- void flag_error (const char ∗msg, const char ∗file, const int line)

  Report an error message.
- void flag_notify (const char ∗msg, const char ∗file, const int line)

  Report a notification.
- FILE ∗ mfopen (const char ∗filename, const char ∗opt)

  Safely open a file.
- vector< double > pbc (const vector< double > coords, const vector< double > box)

  Returns the equivalent cartesian coordinates back in the simulation box assuming periodic boundaries.
- vector< double > pbc (const double ∗coords, const vector< double > box)

  Returns the equivalent cartesian coordinates back in the simulation box assuming periodic boundaries.
- double min_image_dist2 (const vector< double > coords1, const vector< double > coords2, const vector< double > box)

  Return the square of the minimum image distance between 2 coordinate vectors.
- double min_image_dist2 (const Atom ∗a1, const Atom ∗a2, const vector< double > ∗box, double ∗xyz)

  Returns the square of the minimum image distance between 2 atoms, also returns the minimum image distance vector xyz that points from atom1 to atom2.
- double unifRand ()

  Generate a random number between 0 and 1, returns a uniform number in [0,1].
- double unifRand (double a, double b)

  Generate a random number in a real interval.

5.17.1 Detailed Description

Header for Miscellaneous Routines.

Author

Nathan A. Mahynski

5.17.2 Function Documentation

5.17.2.1 void flag_error (const char ∗ *msg,* const char ∗ *file,* const int *line*)

Report an error message.

Error messages are sent to stderr.

Parameters

in	∗*msg*	Character string to print out.
in	∗*file*	**FILE** this function is called from.
in	*line*	**LINE** this function is called from.

5.17.2.2 void flag_notify (const char $*$ *msg,* const char $*$ *file,* const int *line*)

Report a notification.

Notification messages are sent to stderr.

Parameters

`in`	$*$*msg*	Character string to print out.
`in`	$*$*file*	**FILE** this function is called from.
`in`	*line*	**LINE** this function is called from.

5.17.2.3 FILE$*$ mfopen (const char $*$ *filename,* const char $*$ *opt*)

Safely open a file.

Tries to open a file, if it fails it returns a NULL pointer and alerts the user with an error message. If it succeeds it returns the file pointer.

Parameters

`in`	$*$*filename*	Character name of file to open.
`in`	$*$*opt*	File option ("r","w","rw+",etc.).

5.17.2.4 double min_image_dist2 (const vector$<$ double $>$ *coords1,* const vector$<$ double $>$ *coords2,* const vector$<$ double $>$ *box*)

Return the square of the minimum image distance between 2 coordinate vectors.

Parameters

`in`	*coords1*	Vector of cartesian coordinates of one atom
`in`	*coords2*	Vector of cartesian coordinates of the other atom
`in`	*box*	Vector of cartesian coordinates of the box

5.17.2.5 double min_image_dist2 (const **Atom** $*$ *a1,* const **Atom** $*$ *a2,* const vector$<$ double $>$ $*$ *box,* double $*$ *xyz*)

Returns the square of the minimum image distance between 2 atoms, also returns the minimum image distance vector xyz that points from atom1 to atom2.

Parameters

`in`	$*$*a1*	Pointer to one atom
`in`	$*$*a2*	Pointer to the other atom
`in`	*box*	Vector of cartesian coordinates of the box
`in,out`	$*$*xyz*	Array of xyz displacements to be returned to the user (length 3)

5.17.2.6 vector$<$double$>$ pbc (const vector$<$ double $>$ *coords,* const vector$<$ double $>$ *box*)

Returns the equivalent cartesian coordinates back in the simulation box assuming periodic boundaries.

If this routine fails, it returns an empty vector (size = 0).

Parameters

`in`	*coords*	Vector of cartesian coordinates.
`in`	*box*	Vector of box dimensions (L_x, L_y, L_z).

5.17.2.7 vector$<$double$>$ pbc (const double $*$ *coords,* const vector$<$ double $>$ *box*)

Returns the equivalent cartesian coordinates back in the simulation box assuming periodic boundaries.

If this routine fails, it returns an empty vector (size = 0).

Parameters

`in`	$*$*coords*	Array of cartesian coordinates.
`in`	*box*	Vector of box dimensions (L_x, L_y, L_z).

5.17.2.8 double unifRand (double *a,* double *b*)

Generate a random number in a real interval.

Parameters

a	Lower end point of the interval
b	Upper end of the interval

5.18 mpiatom.h File Reference

Variables

- MPI_Datatype MPI_ATOM

  MPI_ATOM is made visible to all routines.

5.18.1 Detailed Description

Author

Nathan A. Mahynski

5.19 read_interaction.cpp File Reference

Source code to read in interaction parameters from a parameter file.

`#include "read_interaction.h"`

Functions

- int read_interactions (const string filename, System $*$sys)

  Function to read in interaction parameters and store them into the system.
- force_energy_ptr get_fn (const string name, vector$<$ double $>$ $*$args, double $*$r_cut_max)

  Function to return a force_energy_ptr given a type of interaction name.

5.19.1 Detailed Description

Source code to read in interaction parameters from a parameter file.

Author

{Nathan A. Mahynski, Carmeline Dsilva}

5.19.2 Function Documentation

5.19.2.1 force_energy_ptr get_fn (const string *name*, vector< double > ∗ *args*, double ∗ *r_cut_max*)

Function to return a force_energy_ptr given a type of interaction name.

Essentially a factory, given a name, return the force_energy_ptr associated with it. Also check the arguments that will be passed to it are in acceptable range.

Parameters

`in`	*name*	Name of interaction, i.e. "fene" or "slj"
`in`	∗*args*	Pointer to vector of arguments that will be passed to this interaction later on
`in,out`	∗*r_cut_max*	Maximum cutoff radius for interactions

5.19.2.2 int read_interactions (const string *filename*, System ∗ *sys*)

Function to read in interaction parameters and store them into the system.

Parameters

`in`	*filename*	name of file containing the interaction parameters to be processed
`in,out`	∗*sys*	Pointer to system

5.20 read_interaction.h File Reference

Header file for reading interactions in.

```
#include <boost/algorithm/string.hpp>
#include <boost/bind.hpp>
#include <boost/function.hpp>
#include "mpi.h"
#include "misc.h"
#include "atom.h"
#include "system.h"
#include "global.h"
```

Functions

- int read_interactions (const string filename, System ∗sys)

  Function to read in interaction parameters and store them into the system.
- force_energy_ptr get_fn (const string name, vector< double > ∗args, double ∗r_cut_max)

  Function to return a force_energy_ptr given a type of interaction name.

5.20.1 Detailed Description

Header file for reading interactions in.

5.20.2 Function Documentation

5.20.2.1 force_energy_ptr get_fn (const string *name*, vector< double > ∗ *args*, double ∗ *r_cut_max*)

Function to return a force_energy_ptr given a type of interaction name.

Essentially a factory, given a name, return the force_energy_ptr associated with it. Also check the arguments that will be passed to it are in acceptable range.

Parameters

`in`	*name*	Name of interaction, i.e. "fene" or "slj"
`in`	∗*args*	Pointer to vector of arguments that will be passed to this interaction later on
`in,out`	∗*r_cut_max*	Maximum cutoff radius for interactions

5.20.2.2 int read_interactions (const string *filename*, System ∗ *sys*)

Function to read in interaction parameters and store them into the system.

Parameters

`in`	*filename*	name of file containing the interaction parameters to be processed
`in,out`	∗*sys*	Pointer to system

5.21 read_xml.cpp File Reference

I/O for XML file format.

`#include "read_xml.h"`

Functions

- int read_xml (const string filename, System ∗sys)

  Read an xml file to initialize a System object.
- int print_xml (const string filename, const System ∗sys)

  Print the current state of the system to a .xml file.
- int write_xyz (const string filename, const System ∗sys, const int timestep, const bool wrap_pos)

  Write an animation file (.xyz)

5.21.1 Detailed Description

I/O for XML file format.

Author

Nathan A. Mahynski

5.21.2 Function Documentation

5.21.2.1 int print_xml (const string *filename*, const System ∗ *sys*)

Print the current state of the system to a .xml file.

Print atom information to an xml file. Returns SAFE_EXIT if successful, else an error flag if failure. This only operates on the master node when multiple processors are used, the rest pause and are sequentially informed to send information as needed.

Parameters

`in`	*filename*	Name of file to open and read.
`in,out`	∗*sys*	Pointer to System object for the main node to stores its information at.

5.21.2.2 int read_xml (const string *filename*, System ∗ *sys*)

Read an xml file to initialize a System object.

Parse an XML file to obtain atom information. Initializes a System object but only stores information that belongs to this processor rank according to domain decomposition. Returns SAFE_EXIT if successful, else returns an error flag.

Parameters

`in`	*filename*	Name of file to open and read.
`in,out`	∗*sys*	Pointer to System object to store its information at.

5.21.2.3 int write_xyz (const string *filename*, const System ∗ *sys*, const int *timestep*, const bool *wrap_pos*)

Write an animation file (.xyz)

Write an xyz file that stores the coordinates of all the atoms in the simulation If timestep=0, then the file is created, or overwritten if it exists already. If timestep>0, then the current information is appended to the existing file. This file can then be read by VMD or another visualization program to produce animations.

Parameters

`in`	*filename*	Name of file to open and write to.
`in`	∗*sys*	System object where the atoms are stored
`in`	*timestep*	Current timestep of the simulation
`in`	*wrap_pos*	Whether the positions should be written in unwrapped (wrap_pos=false) or wrapped (wrap_pos=true) coordinates

5.22 read_xml.h File Reference

I/O for XML.

```
#include <boost/algorithm/string.hpp>
#include <boost/bind.hpp>
#include <boost/function.hpp>
#include "mpi.h"
#include "misc.h"
#include "atom.h"
#include "system.h"
#include "global.h"
#include "read_interaction.h"
```

Functions

- int read_xml (const string filename, System ∗sys)

  Read an xml file to initialize a System object.
- int print_xml (const string filename, const System ∗sys)

  Print the current state of the system to a .xml file.
- int write_xyz (const string filename, const System ∗sys, const int timestep, const bool wrap_pos)

  Write an animation file (.xyz)

5.22.1 Detailed Description

I/O for XML.

Author

Nathan A. Mahynski

5.22.2 Function Documentation

5.22.2.1 int print_xml (const string *filename,* const System ∗ *sys*)

Print the current state of the system to a .xml file.

Print atom information to an xml file. Returns SAFE_EXIT if successful, else an error flag if failure. This only operates on the master node when multiple processors are used, the rest pause and are sequentially informed to send information as needed.

Parameters

`in`	*filename*	Name of file to open and read.
`in,out`	∗*sys*	Pointer to System object for the main node to stores its information at.

5.22.2.2 int read_xml (const string *filename,* System ∗ *sys*)

Read an xml file to initialize a System object.

Parse an XML file to obtain atom information. Initializes a System object but only stores information that belongs to this processor rank according to domain decomposition. Returns SAFE_EXIT if successful, else returns an error flag.

Parameters

`in`	*filename*	Name of file to open and read.
`in,out`	∗*sys*	Pointer to System object to store its information at.

5.22.2.3 int write_xyz (const string *filename,* const System ∗ *sys,* const int *timestep,* const bool *wrap_pos*)

Write an animation file (.xyz)

Write an xyz file that stores the coordinates of all the atoms in the simulation If timestep=0, then the file is created, or overwritten if it exists already. If timestep>0, then the current information is appended to the existing file. This file can then be read by VMD or another visualization program to produce animations.

Parameters

`in`	*filename*	Name of file to open and write to.
`in`	*∗sys*	System object where the atoms are stored
`in`	*timestep*	Current timestep of the simulation
`in`	*wrap_pos*	Whether the positions should be written in unwrapped (wrap_pos=false) or wrapped (wrap_pos=true) coordinates

5.23 system.cpp File Reference

Source code for the System object.

```
#include "system.h"
```

5.23.1 Detailed Description

Source code for the System object.

Author

Nathan A. Mahynski

5.24 system.h File Reference

Header for MD System Information.

```
#include <map>
#include "atom.h"
#include "misc.h"
#include "interaction.h"
#include <list>
#include <algorithm>
#include "global.h"
```

Classes

- class System

5.24.1 Detailed Description

Header for MD System Information.

Author

Nathan A. Mahynski

5.25 verlet.cpp File Reference

Driver for MPI Version of CBEMD verlet.

```
#include "CBEMD.h"
```

Functions

- int main (int argc, char ∗argv[ ])

5.25.1 Detailed Description

Driver for MPI Version of CBEMD verlet.

Authors

{Nathan A. Mahynski, Carmeline Dsilva, Arun L. Prabhu, George Khoury, Frank Ricci, Jun Park}

5.25.2 Function Documentation

5.25.2.1 int main (int *argc,* char ∗ *argv[ ]*)

Parameters

∗*argv[ ]*	./verlet nsteps dt xml_file energy_file animation_file

Index

abort_mpi
    initialize.cpp, 31
add_atom_type
    System, 13
add_atoms
    System, 13
add_bond
    System, 14
add_bond_type
    System, 14
add_ghost_atoms
    System, 14
Andersen, 7
    Andersen, 7
    step, 8
andersen.cpp, 19
    main, 19
Atom, 8
atom.cpp, 19
    create_MPI_ATOM, 20
    delete_MPI_atom, 20
atom.h, 20
    create_MPI_ATOM, 21
    delete_MPI_atom, 21
atom_name
    System, 15
atom_type
    System, 15

bond_name
    System, 15
bond_type
    System, 15

CBEMD.h, 21
clear_ghost_atoms
    System, 15
common.h, 22
communicate_skin_atoms
    domain_decomp.cpp, 23
    domain_decomp.h, 26
create_MPI_ATOM
    atom.cpp, 20
    atom.h, 21

delete_MPI_atom
    atom.cpp, 20
    atom.h, 21
delete_atoms
    System, 15
domain_decomp.cpp, 22
    communicate_skin_atoms, 23
    factorize, 23
    gen_goes_to, 23
    gen_send_lists, 23
    gen_send_table, 23
    gen_sets, 24
    gen_skin_cutoff, 24
    get_processor, 24
    get_xyz_ids, 24
    init_domain_decomp, 25
    power, 25
domain_decomp.h, 25
    communicate_skin_atoms, 26
    factorize, 26
    gen_goes_to, 26
    gen_send_lists, 27
    gen_send_table, 27
    gen_sets, 27
    get_processor, 27
    get_xyz_ids, 27
    init_domain_decomp, 28
    power, 28

end_mpi
    initialize.cpp, 31

factorize
    domain_decomp.cpp, 23
    domain_decomp.h, 26
fene
    interaction.cpp, 34
    interaction.h, 36
FeneException, 9
flag_error
    misc.cpp, 38
    misc.h, 40
flag_notify
    misc.cpp, 38
    misc.h, 40
force_calc
    force_calc.cpp, 29
    force_calc.h, 29
force_calc.cpp, 28
    force_calc, 29
    send_atoms, 29
force_calc.h, 29
    force_calc, 29
    send_atoms, 30

gen_domain_info
  System, 16
gen_goes_to
  domain_decomp.cpp, 23
  domain_decomp.h, 26
gen_send_lists
  domain_decomp.cpp, 23
  domain_decomp.h, 27
gen_send_table
  domain_decomp.cpp, 23
  domain_decomp.h, 27
gen_sets
  domain_decomp.cpp, 24
  domain_decomp.h, 27
gen_skin_cutoff
  domain_decomp.cpp, 24
get_fn
  read_interaction.cpp, 43
  read_interaction.h, 44
get_processor
  domain_decomp.cpp, 24
  domain_decomp.h, 27
get_xyz_ids
  domain_decomp.cpp, 24
  domain_decomp.h, 27
global.h, 30

harmonic
  interaction.cpp, 34
  interaction.h, 36

init_domain_decomp
  domain_decomp.cpp, 25
  domain_decomp.h, 28
initialize.cpp, 31
  abort_mpi, 31
  end_mpi, 31
  initialize_from_files, 31
  start_mpi, 32
initialize_from_files
  initialize.cpp, 31
Integrator, 9
integrator.cpp, 32
  run, 32
integrator.h, 33
  run, 33
Interaction, 10
interaction.cpp, 33
  fene, 34
  harmonic, 34
  slj, 35
interaction.h, 35
  fene, 36
  harmonic, 36
  slj, 37

main
  andersen.cpp, 19
  verlet.cpp, 48
mfopen
  misc.cpp, 38
  misc.h, 41
min_image_dist2
  misc.cpp, 38, 39
  misc.h, 41
misc.cpp, 37
  flag_error, 38
  flag_notify, 38
  mfopen, 38
  min_image_dist2, 38, 39
  pbc, 39
  unifRand, 39
misc.h, 40
  flag_error, 40
  flag_notify, 40
  mfopen, 41
  min_image_dist2, 41
  pbc, 41, 42
  unifRand, 42
mpiatom.h, 42

pbc
  misc.cpp, 39
  misc.h, 41, 42
power
  domain_decomp.cpp, 25
  domain_decomp.h, 28
print_xml
  read_xml.cpp, 45
  read_xml.h, 46

read_interaction.cpp, 42
  get_fn, 43
  read_interactions, 43
read_interaction.h, 43
  get_fn, 44
  read_interactions, 44
read_interactions
  read_interaction.cpp, 43
  read_interaction.h, 44
read_xml
  read_xml.cpp, 45
  read_xml.h, 46
read_xml.cpp, 44
  print_xml, 45
  read_xml, 45
  write_xyz, 45
read_xml.h, 45
  print_xml, 46
  read_xml, 46
  write_xyz, 46
run
  integrator.cpp, 32
  integrator.h, 33

send_atoms
  force_calc.cpp, 29
  force_calc.h, 30

slj
    interaction.cpp, 35
    interaction.h, 37
SljException, 10
start_mpi
    initialize.cpp, 32
step
    Andersen, 8
    Verlet, 16
System, 11
    add_atom_type, 13
    add_atoms, 13
    add_bond, 14
    add_bond_type, 14
    add_ghost_atoms, 14
    atom_name, 15
    atom_type, 15
    bond_name, 15
    bond_type, 15
    clear_ghost_atoms, 15
    delete_atoms, 15
    gen_domain_info, 16
    System, 13
system.cpp, 47
system.h, 47

unifRand
    misc.cpp, 39
    misc.h, 42

Verlet, 16
    step, 16
    Verlet, 16
verlet.cpp, 47
    main, 48

write_xyz
    read_xml.cpp, 45
    read_xml.h, 46