

MD Simulation Code for Sodium Chloride Simulation

Generated by Doxygen 1.8.11

Fri Dec 18 2015 20:35:01

Contents

| | | |
|----------|--|----------|
| 1 | Class Index | 1 |
| 1.1 | Class List | 1 |
| 2 | Class Documentation | 3 |
| 2.1 | CAtom Class Reference | 3 |
| 2.1.1 | Detailed Description | 3 |
| 2.1.2 | Constructor & Destructor Documentation | 3 |
| 2.1.2.1 | CAtom(CVector3< double > a_position, const int a_atomindex, const int a_↵ atomtype) | 3 |
| 2.1.2.2 | CAtom(CVector3< double > a_position, CVector3< double > a_vel, const int a_atomindex, const int a_atomtype) | 3 |
| 2.1.3 | Member Function Documentation | 4 |
| 2.1.3.1 | Display() | 4 |
| 2.1.3.2 | getAtomIndex() | 4 |
| 2.1.3.3 | getAtomtype() | 4 |
| 2.1.3.4 | getCoord() | 4 |
| 2.1.3.5 | getForces() | 4 |
| 2.1.3.6 | getVelocity() | 4 |
| 2.1.3.7 | setForces(const double) | 4 |
| 2.1.3.8 | setVelocity(CVector3< double > a_vel) | 4 |
| 2.2 | CCubes Class Reference | 4 |
| 2.2.1 | Detailed Description | 5 |
| 2.2.2 | Member Function Documentation | 5 |
| 2.2.2.1 | getCubeCenter() | 5 |
| 2.2.2.2 | getInteriorMolecule() | 5 |
| 2.2.2.3 | setCubeSize(double a_cubeSize) | 5 |
| 2.3 | CForcefield Class Reference | 5 |
| 2.3.1 | Constructor & Destructor Documentation | 5 |
| 2.3.1.1 | CForcefield(const char *) | 5 |
| 2.3.2 | Member Function Documentation | 5 |
| 2.3.2.1 | display() | 5 |
| 2.3.2.2 | getCharges(const int i) | 5 |

| | | |
|---------|--|----|
| 2.3.2.3 | getEps(const int a_value1, const int a_value2) | 6 |
| 2.3.2.4 | getMass() | 6 |
| 2.3.2.5 | getMassAtom(const int i) | 6 |
| 2.3.2.6 | getSigma2(const int a_value1, const int a_value2) | 6 |
| 2.4 | CMolecule Class Reference | 6 |
| 2.4.1 | Detailed Description | 6 |
| 2.4.2 | Member Function Documentation | 6 |
| 2.4.2.1 | Display() | 6 |
| 2.4.2.2 | getActualAtom(const int a_index) | 6 |
| 2.4.2.3 | getAtom(const int a_index) | 7 |
| 2.4.2.4 | getNatom() | 7 |
| 2.4.2.5 | operator[](const int a_index) | 7 |
| 2.4.2.6 | setMolIndex(int index) | 7 |
| 2.4.2.7 | ShiftCoord(CVector3< double > &) | 7 |
| 2.5 | CMoves Class Reference | 7 |
| 2.5.1 | Detailed Description | 7 |
| 2.5.2 | Constructor & Destructor Documentation | 7 |
| 2.5.2.1 | ~CMoves() | 7 |
| 2.5.3 | Member Function Documentation | 7 |
| 2.5.3.1 | calForce(CSystem &, CForcefield &, CParam &, int timeStep) | 7 |
| 2.5.3.2 | propagate(CSystem &, CForcefield &, CParam &) | 8 |
| 2.6 | CNonbonded Class Reference | 8 |
| 2.6.1 | Detailed Description | 8 |
| 2.6.2 | Member Function Documentation | 8 |
| 2.6.2.1 | callInterNonbondedNaive(CSystem &, CForcefield &, CParam &) | 8 |
| 2.6.2.2 | callInterNonbondedNblast(CSystem &, CForcefield &, CParam &) | 8 |
| 2.6.2.3 | callInterNonbondedPME(CSystem &, CForcefield &, CParam &) | 8 |
| 2.7 | CParam Class Reference | 9 |
| 2.7.1 | Constructor & Destructor Documentation | 9 |
| 2.7.1.1 | CParam() | 9 |
| 2.7.1.2 | CParam(char *) | 9 |
| 2.7.2 | Member Function Documentation | 9 |
| 2.7.2.1 | createDirectionUnitVector() | 9 |
| 2.7.2.2 | Display() | 10 |
| 2.7.2.3 | getCELLX() | 10 |
| 2.7.2.4 | getCELLY() | 10 |
| 2.7.2.5 | getCELLZ() | 10 |
| 2.7.2.6 | getDt() | 10 |
| 2.7.2.7 | getKinstep() | 10 |
| 2.7.2.8 | getMdoutfreq() | 10 |

| | | |
|----------|--|----|
| 2.7.2.9 | getMethod() | 10 |
| 2.7.2.10 | getMoviefilefreq() | 10 |
| 2.7.2.11 | getMovieflag() | 10 |
| 2.7.2.12 | getMoviefname() | 10 |
| 2.7.2.13 | getNatomEachside() | 10 |
| 2.7.2.14 | getParmfile() | 11 |
| 2.7.2.15 | getPBCS() | 11 |
| 2.7.2.16 | getReadpos() | 11 |
| 2.7.2.17 | getReadPosfname() | 11 |
| 2.7.2.18 | getReadVelfname() | 11 |
| 2.7.2.19 | getRestartfreq() | 11 |
| 2.7.2.20 | getRestartPosfname() | 11 |
| 2.7.2.21 | getRestartVel() | 11 |
| 2.7.2.22 | getRestartVelfname() | 11 |
| 2.7.2.23 | getTemp() | 11 |
| 2.7.2.24 | getVDWcutoff() | 11 |
| 2.8 | CSystem Class Reference | 12 |
| 2.8.1 | Detailed Description | 12 |
| 2.8.2 | Constructor & Destructor Documentation | 12 |
| 2.8.2.1 | CSystem() | 12 |
| 2.8.2.2 | CSystem(CParam &, CForcefield &) | 12 |
| 2.8.3 | Member Function Documentation | 12 |
| 2.8.3.1 | assignParticlesToCubes(CParam &a_params) | 12 |
| 2.8.3.2 | createCubes(CParam &a_params) | 12 |
| 2.8.3.3 | createLattice(const double, const double, const double, const int) | 13 |
| 2.8.3.4 | display() | 13 |
| 2.8.3.5 | fullDisplay() | 13 |
| 2.8.3.6 | GetCM(CVector3< double > &, vector< double > &a_mass) | 13 |
| 2.8.3.7 | getIndex(const Point &a_pt, Point &m_lowCorner, Point &m_highCorner) const | 13 |
| 2.8.3.8 | getMol(const int index) | 13 |
| 2.8.3.9 | getNmols() | 13 |
| 2.8.3.10 | getPoint(int k, Point &m_lowCorner, Point &m_highCorner) const | 13 |
| 2.8.3.11 | initializeVelocity(const double, vector< double > &) | 13 |
| 2.8.3.12 | readPosition(const char *) | 13 |
| 2.8.3.13 | readVelocity(const char *) | 13 |
| 2.8.3.14 | setForcesZero() | 13 |
| 2.8.3.15 | ShiftCoord(CForcefield &) | 14 |
| 2.8.3.16 | writeMovie(ofstream &, int) | 14 |
| 2.8.3.17 | writeRestartpos(const char *) | 14 |
| 2.8.3.18 | writeRestartvel(const char *) | 14 |

| | | |
|--------------|--|-----------|
| 2.9 | CVector3< X > Class Template Reference | 14 |
| 2.9.1 | Constructor & Destructor Documentation | 14 |
| 2.9.1.1 | CVector3(const X a_value) | 14 |
| 2.9.2 | Member Function Documentation | 14 |
| 2.9.2.1 | copy(CVector3 &) | 14 |
| 2.9.2.2 | Display() | 15 |
| 2.9.2.3 | minImage(CVector3 &a_halfbox, CVector3 &a_boxsize) | 15 |
| 2.9.2.4 | operator*(double a_const) | 15 |
| 2.9.2.5 | operator*=(const double) | 15 |
| 2.9.2.6 | operator+(CVector3 &a_vec) | 15 |
| 2.9.2.7 | operator+=(CVector3 &) | 15 |
| 2.9.2.8 | operator-(CVector3 &a_vec) | 15 |
| 2.9.2.9 | operator-=(CVector3 &) | 15 |
| 2.9.2.10 | operator/(double a_const) | 15 |
| 2.9.2.11 | operator/=(const double) | 15 |
| 2.9.2.12 | operator[](const int) | 15 |
| 2.9.2.13 | SqLength() const | 15 |
| Index | | 17 |

Chapter 1

Class Index

1.1 Class List

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

| | |
|-------------------------------------|----|
| CAtom | 3 |
| CCubes | 4 |
| CForcefield | 5 |
| CMolecule | 6 |
| CMoves | 7 |
| CNonbonded | 8 |
| CParam | 9 |
| CSystem | 12 |
| CVector3< X > | 14 |

Chapter 2

Class Documentation

2.1 CAtom Class Reference

```
#include <atom.h>
```

Public Member Functions

- [CAtom](#) ([CVector3](#)< double > a_position, const int a_atomindex, const int a_atomtype)
- [CAtom](#) ([CVector3](#)< double > a_position, [CVector3](#)< double > a_vel, const int a_atomindex, const int a_atomtype)
- void [Display](#) ()
- void [setVelocity](#) ([CVector3](#)< double > a_vel)
- void [setForces](#) (const double)
- [CVector3](#)< double > & [getVelocity](#) ()
- [CVector3](#)< double > & [getCoord](#) ()
- [CVector3](#)< double > & [getForces](#) ()
- const int [getAtomtype](#) ()
- const int [getAtomIndex](#) ()

2.1.1 Detailed Description

The atom class holds all the properties of an atom.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 CAtom::CAtom ([CVector3](#)< double > a_position, const int a_atomindex, const int a_atomtype)

Default constructor for the atom Atom constructor given the positions and atom information. Velocity will be randomized.

2.1.2.2 CAtom::CAtom ([CVector3](#)< double > a_position, [CVector3](#)< double > a_vel, const int a_atomindex, const int a_atomtype)

Atom constructor given both the positions and velocities, as well as the atom information

2.1.3 Member Function Documentation

2.1.3.1 void CAtom::Display ()

Writes out all the properties of the atom, i.e. position, velocity, forces

2.1.3.2 const int CAtom::getAtomIndex () [inline]

Returns the atom index

2.1.3.3 const int CAtom::getAtomtype () [inline]

Returns the atom type

2.1.3.4 CVector3<double>& CAtom::getCoord () [inline]

Returns the coordinates vector of the atom

2.1.3.5 CVector3<double>& CAtom::getForces () [inline]

Returns the forces vector of the atom

2.1.3.6 CVector3<double>& CAtom::getVelocity () [inline]

Returns the velocity vector of the atom

2.1.3.7 void CAtom::setForces (const double a_val)

Sets the forces in all directions to the input value

2.1.3.8 void CAtom::setVelocity (CVector3< double > a_vel)

Sets the x,y,z velocity components of the atom to the x,y,z components of the input velocity vector

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

- atom.h
- atom.cpp

2.2 CCubes Class Reference

```
#include <Cubes.h>
```

Public Member Functions

- double **getCubeSize** ()
- void **setCubeSize** (double a_cubeSize)
- Point & **getCubeCenter** ()
- vector< CMolecule > & **getInteriorMolecule** ()

2.2.1 Detailed Description

This class is an element to store atoms to be able to use neighborlists to calculate interactions within cutoffs.

2.2.2 Member Function Documentation

2.2.2.1 Point& CCubes::getCubeCenter () [inline]

Set the size of the cube

2.2.2.2 vector<CMolecule>& CCubes::getInteriorMolecule () [inline]

Return the center of the cube

2.2.2.3 void CCubes::setCubeSize (double a_cubeSize) [inline]

Return size of the cube

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

- Cubes.h

2.3 CForcefield Class Reference

Public Member Functions

- [CForcefield](#) (const char *)
- void [display](#) ()
- vector< double > & [getMass](#) ()
- const double [getMassAtom](#) (const int i)
- const double [getEps](#) (const int a_value1, const int a_value2)
- const double [getSigma2](#) (const int a_value1, const int a_value2)
- const double [getCharges](#) (const int i)

2.3.1 Constructor & Destructor Documentation

2.3.1.1 CForcefield::CForcefield (const char * a_prm)

Default constructor for force field class object Constructor for force field class object from input configuration file

2.3.2 Member Function Documentation

2.3.2.1 void CForcefield::display ()

Function to write out all the parameters

2.3.2.2 const double CForcefield::getCharges (const int i) [inline]

Returns the charge for a particular atom type

2.3.2.3 `const double CForcefield::getEps (const int a_value1, const int a_value2)` `[inline]`

Returns the combined epsilon value for a particular pair of atom types

2.3.2.4 `vector<double>& CForcefield::getMass ()` `[inline]`

Returns vector of masses

2.3.2.5 `const double CForcefield::getMassAtom (const int i)` `[inline]`

Returns the mass of specific atom type

2.3.2.6 `const double CForcefield::getSigma2 (const int a_value1, const int a_value2)` `[inline]`

Returns the combined sigma value for a particular pair of atom types

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

- forcefield.h
- forcefield.cpp

2.4 C Molecule Class Reference

```
#include <molecule.h>
```

Public Member Functions

- **C Molecule** (const int a_natoms, vector< double > a_tuple, vector< int > a_atomindex, int a_moleculeIndex, vector< int > a_atomtype)
- void [Display](#) ()
- void [ShiftCoord](#) (CVector3< double > &)
- [CAtom](#) & [operator\[\]](#) (const int a_index)
- [CAtom](#) & [getAtom](#) (const int a_index)
- [CAtom](#) [getActualAtom](#) (const int a_index)
- const int [getNatom](#) ()
- const int [getMolIndex](#) ()
- void [setMolIndex](#) (int index)

2.4.1 Detailed Description

The molecule class holds all atoms within a molecule.

2.4.2 Member Function Documentation

2.4.2.1 `void C Molecule::Display ()`

Writes out the molecule

2.4.2.2 `CAtom C Molecule::getActualAtom (const int a_index)` `[inline]`

Returns the actual atom object within the molecule by index

2.4.2.3 `CAtom& CMolecule::getAtom (const int a_index)` `[inline]`

Returns a reference to an atom within the molecule by index

2.4.2.4 `const int CMolecule::getNatom ()` `[inline]`

Returns the number of atoms within the molecule

2.4.2.5 `CAtom& CMolecule::operator[] (const int a_index)` `[inline]`

Returns a reference to an atom within the molecule by index

2.4.2.6 `void CMolecule::setMolIndex (int index)` `[inline]`

Returns the global index of the molecule

2.4.2.7 `void CMolecule::ShiftCoord (CVector3< double > & a_center)`

Shifts the coordinates of the molecules by the vector

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

- molecule.h
- molecule.cpp

2.5 CMoves Class Reference

```
#include <moves.h>
```

Public Member Functions

- [~CMoves](#) ()
- void [propagate](#) (CSystem &, CForcefield &, CParam &)
- void [calForce](#) (CSystem &, CForcefield &, CParam &, int timeStep)
- void [assignCubes](#) (CSystem &)

2.5.1 Detailed Description

This class performs the dynamic update step, and calculates the updated positions and velocities.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 `CMoves::~~CMoves ()` `[inline]`

Default constructor for moves class object

2.5.3 Member Function Documentation

2.5.3.1 `void CMoves::calForce (CSystem & a_sys, CForcefield & a_ff, CParam & a_prm, int timeStep)`

This class calls the appropriate force calculation method based on the keyfile argument

2.5.3.2 void CMoves::propagate (CSystem & a_sys, CForcefield & a_ff, CParam & a_prm)

Default destructor for moves class object This class performs the dynamic update using the Velocity Verlet Algorithm

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

- moves.h
- moves.cpp

2.6 CNonbonded Class Reference

```
#include <Nonbonded.h>
```

Public Member Functions

- void callInterNonbondedNaive (CSystem &, CForcefield &, CParam &)
- void callInterNonbondedNblast (CSystem &, CForcefield &, CParam &)
- void callInterNonbondedPME (CSystem &, CForcefield &, CParam &)

2.6.1 Detailed Description

This class calculates the non bonded forces on the atoms.

2.6.2 Member Function Documentation

2.6.2.1 void CNonbonded::callInterNonbondedNaive (CSystem & a_sys, CForcefield & a_ff, CParam & a_prm)

Calculates the non bonded forces using direct pairwise computation

2.6.2.2 void CNonbonded::callInterNonbondedNblast (CSystem & a_system, CForcefield & a_ff, CParam & a_params)

Calculates the non bonded forces using a neighbor list approach, has to be used in conjunction with the PME method

2.6.2.3 void CNonbonded::callInterNonbondedPME (CSystem & a_system, CForcefield & a_ff, CParam & a_params)

Calculates the non bonded forces using the PME method. The real space forces are calculated via a neighbor list Calling the pairwise for the less than cutoff calculations.

Declare required variables.

Declaring variables required for the 4 step ewald protocol.

Initialize the omegaGrid values to zero.

Deposit the charges on the grid

Convolution with the Green's function using Hockney's algorithm, to obtain the potential on the grid.

Create the grid field container (vector<vector<Real>> in this case) and initialize to zero.

Calculate the fields on the grid points via finite differences.

Compute field only for interior grid points.

Create the particles field container (vector<vector<Real>> in this case) and initialize to zero.

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

- Nonbonded.h
- Nonbonded.cpp

2.7 CParam Class Reference

Public Member Functions

- [CParam](#) ()
- [CParam](#) (char *)
- void [Display](#) ()
- const int [getKinstep](#) ()
- const double [getDt](#) ()
- const bool [getReadpos](#) ()
- const char * [getReadPosfname](#) ()
- const bool [getRestartVel](#) ()
- const char * [getReadVelfname](#) ()
- const double [getCELLX](#) ()
- const double [getCELLY](#) ()
- const double [getCELLZ](#) ()
- const int [getNatomEachside](#) ()
- const double [getTemp](#) ()
- const char * [getParmfile](#) ()
- const double [getVDWcutoff](#) ()
- const bool [getPBCS](#) ()
- const int [getRestartfreq](#) ()
- const int [getMdoutfreq](#) ()
- const int [getMoviefilefreq](#) ()
- const bool [getMovieflag](#) ()
- const char * [getMoviefname](#) ()
- const char * [getRestartPosfname](#) ()
- const char * [getRestartVelfname](#) ()
- const char * [getMethod](#) ()
- void [createDirectionUnitVector](#) ()
- vector< Point > & [getDirectionUnitVector](#) ()

2.7.1 Constructor & Destructor Documentation

2.7.1.1 CParam::CParam ()

Default constructor

2.7.1.2 CParam::CParam (char * a_fname)

Constructor from input parameter file

2.7.2 Member Function Documentation

2.7.2.1 void CParam::createDirectionUnitVector ()

Initializes the direction unit vector for calculating the neighbouring cubes for the nblist

2.7.2.2 void CParam::Display ()

Destructor Writes out the parameters

2.7.2.3 const double CParam::getCELLX () [inline]

Return cell x-dimension

2.7.2.4 const double CParam::getCELLY () [inline]

Return cell x-dimension

2.7.2.5 const double CParam::getCELLZ () [inline]

Return cell x-dimension

2.7.2.6 const double CParam::getDt () [inline]

Return timestep

2.7.2.7 const int CParam::getKinstep () [inline]

Return number of dynamics steps

2.7.2.8 const int CParam::getMdoutfreq () [inline]

Returns frequency for mdout data file

2.7.2.9 const char* CParam::getMethod () [inline]

Returns electrostatic computation method cutoff

2.7.2.10 const int CParam::getMoviefilefreq () [inline]

Returns frequency for writing out movie file

2.7.2.11 const bool CParam::getMovieflag () [inline]

Returns flag for writing out movie file

2.7.2.12 const char* CParam::getMoviefname () [inline]

Returns name of movie file

2.7.2.13 const int CParam::getNatomEachside () [inline]

Return number of atoms along each side of box

2.7.2.14 `const char* CParam::getParmfile () [inline]`

Return parameters file name

2.7.2.15 `const bool CParam::getPBCS () [inline]`

Return periodic boundary conditions flag

2.7.2.16 `const bool CParam::getReadpos () [inline]`

Return read positions flag

2.7.2.17 `const char* CParam::getReadPosfname () [inline]`

Return positions filename

2.7.2.18 `const char* CParam::getReadVelfname () [inline]`

Return velocities filename

2.7.2.19 `const int CParam::getRestartfreq () [inline]`

Returns frequency for writing restart file

2.7.2.20 `const char* CParam::getRestartPosfname () [inline]`

Returns name of restart positions file

2.7.2.21 `const bool CParam::getRestartVel () [inline]`

Return read velocities flag

2.7.2.22 `const char* CParam::getRestartVelfname () [inline]`

Returns name of restart velocities file

2.7.2.23 `const double CParam::getTemp () [inline]`

Return simulation temperature

2.7.2.24 `const double CParam::getVDWcutoff () [inline]`

Return Van der Waal's cutoff

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

- parameters.h
- parameters.cpp

2.8 CSystem Class Reference

```
#include <system.h>
```

Public Member Functions

- [CSystem](#) ()
- [CSystem](#) (CParam &, CForcefield &)
- void [readPosition](#) (const char *)
- void [createLattice](#) (const double, const double, const double, const int)
- void [readVelocity](#) (const char *)
- void [initializeVelocity](#) (const double, vector< double > &)
- void [setForcesZero](#) ()
- void [createCubes](#) (CParam &a_params)
- void [assignParticlesToCubes](#) (CParam &a_params)
- Point [getPoint](#) (int k, Point &m_lowCorner, Point &m_highCorner) const
- int [getindex](#) (const Point &a_pt, Point &m_lowCorner, Point &m_highCorner) const
- vector< CCubes > & [getCubeSet](#) ()
- void [fullDisplay](#) ()
- void [display](#) ()
- void [ShiftCoord](#) (CForcefield &)
- void [GetCM](#) (CVector3< double > &, vector< double > &a_mass)
- void [writeRestartvel](#) (const char *)
- void [writeRestartpos](#) (const char *)
- void [writeMovie](#) (ofstream &, int)
- const int [getNmols](#) ()
- CMolecule & [getMol](#) (const int index)

2.8.1 Detailed Description

This is the entire system class, which contains all the molecules.

2.8.2 Constructor & Destructor Documentation

2.8.2.1 CSystem::CSystem ()

Default constructor

2.8.2.2 CSystem::CSystem (CParam & a_param, CForcefield & a_ff)

Constructor based on the parameter and forcefield files

2.8.3 Member Function Documentation

2.8.3.1 void CSystem::assignParticlesToCubes (CParam & a_params)

Assigns particles/atoms to cubes

2.8.3.2 void CSystem::createCubes (CParam & a_params)

Creates cubes for neighbor list assignment

2.8.3.3 void CSystem::createLattice (const double *a_cellx*, const double *a_celly*, const double *a_cellz*, const int *a_natom_eachside*)

Creates a default lattice if no input file is specified

2.8.3.4 void CSystem::display ()

Writes out a note about the system information

2.8.3.5 void CSystem::fullDisplay ()

Returns the entire cubeseet Writes out the entire system

2.8.3.6 void CSystem::GetCM (CVector3< double > & *a_center*, vector< double > & *a_mass*)

Calculates the coordinates of the system center of mass

2.8.3.7 int CSystem::getIndex (const Point & *a_pt*, Point & *m_lowCorner*, Point & *m_highCorner*) const

Returns index by point

2.8.3.8 CMolecule& CSystem::getMol (const int *index*) [inline]

Returns a molecule based on index

2.8.3.9 const int CSystem::getNmols () [inline]

Default destructor Returns the number of molecules in the system

2.8.3.10 Point CSystem::getPoint (int *k*, Point & *m_lowCorner*, Point & *m_highCorner*) const

Returns point by index

2.8.3.11 void CSystem::initializeVelocity (const double *a_Temp*, vector< double > & *a_mass*)

Initializes random velocities from a Gaussian if no input velocities file is specified

2.8.3.12 void CSystem::readPosition (const char * *a_posfname*)

Reads positions from the input coordinate file

2.8.3.13 void CSystem::readVelocity (const char * *a_velfname*)

Reads velocities from the input velocities file

2.8.3.14 void CSystem::setForcesZero ()

Sets the forces on all atom to zero

2.8.3.15 void CSystem::ShiftCoord (CForcefield & a_ff)

Shifts the coordinates based to center the system

2.8.3.16 void CSystem::writeMovie (ofstream & a_file, int a_time)

Writes the trajectory for visualization

2.8.3.17 void CSystem::writeRestartpos (const char * a_posfname)

Writes coordinates for restarting the system

2.8.3.18 void CSystem::writeRestartvel (const char * a_velfname)

Writes velocities for restarting the system

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

- system.h
- system.cpp

2.9 CVector3< X > Class Template Reference

Public Member Functions

- [CVector3](#) (const X a_value)
- X & [operator\[\]](#) (const int)
- void [Display](#) ()
- void [copy](#) (CVector3 &)
- void [operator+=](#) (CVector3 &)
- void [operator-=](#) (CVector3 &)
- void [operator/=](#) (const double)
- void [operator*=](#) (const double)
- [CVector3 operator*](#) (double a_const)
- [CVector3 operator/](#) (double a_const)
- [CVector3 operator-](#) (CVector3 &a_vec)
- [CVector3 operator+](#) (CVector3 &a_vec)
- void [minImage](#) (CVector3 &a_halfbox, CVector3 &a_boxsize)
- double [SqlLength](#) () const

2.9.1 Constructor & Destructor Documentation

2.9.1.1 template<class X> CVector3< X >::CVector3 (const X a_value)

Default constructor Constructor with input value

2.9.2 Member Function Documentation

2.9.2.1 template<class X> void CVector3< X >::copy (CVector3< X > & a_vector)

Copy constructor

2.9.2.2 `template<class X> void CVector3< X >::Display ()`

Writes out the object tuple

2.9.2.3 `template<class X> void CVector3< X >::minImage (CVector3< X > & a_halfbox, CVector3< X > & a_boxsize)`

Compute Minimum Image

2.9.2.4 `template<class X> CVector3< X > CVector3< X >::operator* (double a_const)`

Return Scalar Multiplication Operator

2.9.2.5 `template<class X> void CVector3< X >::operator*= (const double a)`

Internal Scalar Multiplication Operator

2.9.2.6 `template<class X> CVector3< X > CVector3< X >::operator+ (CVector3< X > & a_vec)`

Return Vector Addition Operator

2.9.2.7 `template<class X> void CVector3< X >::operator+= (CVector3< X > & a_v)`

Internal Vector Addition Operator

2.9.2.8 `template<class X> CVector3< X > CVector3< X >::operator- (CVector3< X > & a_vec)`

Return Vector Subtraction Operator

2.9.2.9 `template<class X> void CVector3< X >::operator-= (CVector3< X > & a_v)`

Internal Vector Subtraction Operator

2.9.2.10 `template<class X> CVector3< X > CVector3< X >::operator/ (double a_const)`

Return Scalar Division Operator

2.9.2.11 `template<class X> void CVector3< X >::operator/= (const double a)`

Internal Scalar Division Operator

2.9.2.12 `template<class X> X & CVector3< X >::operator[] (const int a_index)`

Access operator to access the desired element from the internal tuple

2.9.2.13 `template<class X> double CVector3< X >::SqLength () const`

Calculate Magnitude of Vector

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

- `vector3.h`
- `vector3Implem.h`

Index

- ~CMoves
 - CMoves, [7](#)
- assignParticlesToCubes
 - CSystem, [12](#)
- CAtom, [3](#)
 - CAtom, [3](#)
 - Display, [4](#)
 - getAtomIndex, [4](#)
 - getAtomtype, [4](#)
 - getCoord, [4](#)
 - getForces, [4](#)
 - getVelocity, [4](#)
 - setForces, [4](#)
 - setVelocity, [4](#)
- CCubes, [4](#)
 - getCubeCenter, [5](#)
 - getInteriorMolecule, [5](#)
 - setCubeSize, [5](#)
- CForcefield, [5](#)
 - CForcefield, [5](#)
 - display, [5](#)
 - getCharges, [5](#)
 - getEps, [5](#)
 - getMass, [6](#)
 - getMassAtom, [6](#)
 - getSigma2, [6](#)
- CMolecule, [6](#)
 - Display, [6](#)
 - getActualAtom, [6](#)
 - getAtom, [6](#)
 - getNatom, [7](#)
 - operator[], [7](#)
 - setMolIndex, [7](#)
 - ShiftCoord, [7](#)
- CMoves, [7](#)
 - ~CMoves, [7](#)
 - calForce, [7](#)
 - propagate, [7](#)
- CNonbonded, [8](#)
 - callInterNonbondedNaive, [8](#)
 - callInterNonbondedNblist, [8](#)
 - callInterNonbondedPME, [8](#)
- CParam, [9](#)
 - CParam, [9](#)
 - createDirectionUnitVector, [9](#)
 - Display, [9](#)
 - getCELLX, [10](#)
 - getCELLY, [10](#)
 - getCELLZ, [10](#)
 - getDt, [10](#)
 - getKinstep, [10](#)
 - getMdoutfreq, [10](#)
 - getMethod, [10](#)
 - getMoviefilefreq, [10](#)
 - getMovieflag, [10](#)
 - getMoviefname, [10](#)
 - getNatomEachside, [10](#)
 - getPBCS, [11](#)
 - getParmfile, [10](#)
 - getReadPosfname, [11](#)
 - getReadVelfname, [11](#)
 - getReadpos, [11](#)
 - getRestartPosfname, [11](#)
 - getRestartVel, [11](#)
 - getRestartVelfname, [11](#)
 - getRestartfreq, [11](#)
 - getTemp, [11](#)
 - getVDWcutoff, [11](#)
- CSystem, [12](#)
 - assignParticlesToCubes, [12](#)
 - CSystem, [12](#)
 - createCubes, [12](#)
 - createLattice, [12](#)
 - display, [13](#)
 - fullDisplay, [13](#)
 - GetCM, [13](#)
 - getMol, [13](#)
 - getNmols, [13](#)
 - getPoint, [13](#)
 - getindex, [13](#)
 - initializeVelocity, [13](#)
 - readPosition, [13](#)
 - readVelocity, [13](#)
 - setForcesZero, [13](#)
 - ShiftCoord, [13](#)
 - writeMovie, [14](#)
 - writeRestartpos, [14](#)
 - writeRestartvel, [14](#)
- CVector3
 - CVector3, [14](#)
 - copy, [14](#)
 - Display, [14](#)
 - minImage, [15](#)
 - operator*, [15](#)
 - operator*=[15](#)
 - operator+, [15](#)
 - operator+=, [15](#)

- operator-, 15
- operator=, 15
- operator/, 15
- operator/=: 15
- operator[], 15
- SqLength, 15
- CVector3< X >, 14
- calForce
 - CMoves, 7
- callInterNonbondedNaive
 - CNonbonded, 8
- callInterNonbondedNblast
 - CNonbonded, 8
- callInterNonbondedPME
 - CNonbonded, 8
- copy
 - CVector3, 14
- createCubes
 - CSystem, 12
- createDirectionUnitVector
 - CParam, 9
- createLattice
 - CSystem, 12
- Display
 - CAtom, 4
 - CMolecule, 6
 - CParam, 9
 - CVector3, 14
- display
 - CForcefield, 5
 - CSystem, 13
- fullDisplay
 - CSystem, 13
- getActualAtom
 - CMolecule, 6
- getAtom
 - CMolecule, 6
- getAtomIndex
 - CAtom, 4
- getAtomtype
 - CAtom, 4
- getCELLX
 - CParam, 10
- getCELLY
 - CParam, 10
- getCELLZ
 - CParam, 10
- getCharges
 - CForcefield, 5
- GetCM
 - CSystem, 13
- getCoord
 - CAtom, 4
- getCubeCenter
 - CCubes, 5
- getDt
 - CParam, 10
- getEps
 - CForcefield, 5
- getForces
 - CAtom, 4
- getInteriorMolecule
 - CCubes, 5
- getKinstep
 - CParam, 10
- getMass
 - CForcefield, 6
- getMassAtom
 - CForcefield, 6
- getMdoutfreq
 - CParam, 10
- getMethod
 - CParam, 10
- getMol
 - CSystem, 13
- getMoviefilefreq
 - CParam, 10
- getMovieflag
 - CParam, 10
- getMoviefname
 - CParam, 10
- getNatom
 - CMolecule, 7
- getNatomEachside
 - CParam, 10
- getNmols
 - CSystem, 13
- getPBCS
 - CParam, 11
- getParmfile
 - CParam, 10
- getPoint
 - CSystem, 13
- getReadPosfname
 - CParam, 11
- getReadVelfname
 - CParam, 11
- getReadpos
 - CParam, 11
- getRestartPosfname
 - CParam, 11
- getRestartVel
 - CParam, 11
- getRestartVelfname
 - CParam, 11
- getRestartfreq
 - CParam, 11
- getSigma2
 - CForcefield, 6
- getTemp
 - CParam, 11
- getVDWcutoff
 - CParam, 11
- getVelocity

- CAtom, [4](#)
- getIndex
 - CSystem, [13](#)
- initializeVelocity
 - CSystem, [13](#)
- minImage
 - CVector3, [15](#)
- operator*
 - CVector3, [15](#)
- operator*=
 - CVector3, [15](#)
- operator+
 - CVector3, [15](#)
- operator+=
 - CVector3, [15](#)
- operator-
 - CVector3, [15](#)
- operator-=
 - CVector3, [15](#)
- operator/
 - CVector3, [15](#)
- operator/=
 - CVector3, [15](#)
- operator[]
 - CMolecule, [7](#)
 - CVector3, [15](#)
- propagate
 - CMoves, [7](#)
- readPosition
 - CSystem, [13](#)
- readVelocity
 - CSystem, [13](#)
- setCubeSize
 - CCubes, [5](#)
- setForces
 - CAtom, [4](#)
- setForcesZero
 - CSystem, [13](#)
- setMolIndex
 - CMolecule, [7](#)
- setVelocity
 - CAtom, [4](#)
- ShiftCoord
 - CMolecule, [7](#)
 - CSystem, [13](#)
- SqLength
 - CVector3, [15](#)
- writeMovie
 - CSystem, [14](#)
- writeRestartpos
 - CSystem, [14](#)
- writeRestartvel
 - CSystem, [14](#)