

Molecular Dynamics Code for Noble Gases
1.0

Generated by Doxygen 1.9.8

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 functions.c File Reference	3
2.1.1 Function Documentation	4
2.1.1.1 allocate_2d_array()	4
2.1.1.2 calculate_accelerations()	4
2.1.1.3 calculate_distances()	5
2.1.1.4 calculate_kinetic_energy()	6
2.1.1.5 calculate_potential_energy()	6
2.1.1.6 calculate_total_energy()	6
2.1.1.7 check_energy()	7
2.1.1.8 free_2d_array()	7
2.1.1.9 open_output()	8
2.1.1.10 print_output()	8
2.1.1.11 read_coords_and_masses()	9
2.1.1.12 read_natoms()	9
2.1.1.13 update_positions()	9
2.1.1.14 update_velocities()	10
2.1.1.15 validate_atoms()	10
2.2 headers.h File Reference	11
2.2.1 Macro Definition Documentation	12
2.2.1.1 ARGON_EPSILON	12
2.2.1.2 ARGON_MASS	12
2.2.1.3 ARGON_SIGMA	12
2.2.1.4 CONSTANTS_H	12
2.2.2 Function Documentation	12
2.2.2.1 allocate_2d_array()	12
2.2.2.2 calculate_accelerations()	13
2.2.2.3 calculate_distances()	13
2.2.2.4 calculate_kinetic_energy()	14
2.2.2.5 calculate_potential_energy()	14
2.2.2.6 calculate_total_energy()	15
2.2.2.7 check_energy()	15
2.2.2.8 free_2d_array()	15
2.2.2.9 open_output()	17
2.2.2.10 print_output()	17
2.2.2.11 read_coords_and_masses()	18
2.2.2.12 read_natoms()	18
2.2.2.13 update_positions()	19
2.2.2.14 update_velocities()	19

2.2.2.15 validate_atoms()	19
2.3 headers.h	20
2.4 main.c File Reference	20
2.4.1 Function Documentation	21
2.4.1.1 main()	21
Index	23

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

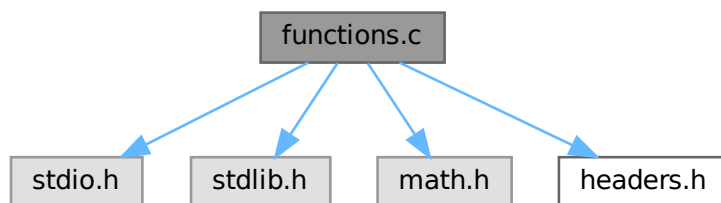
functions.c	3
headers.h	11
main.c	20

Chapter 2

File Documentation

2.1 functions.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "headers.h"
Include dependency graph for functions.c:
```



Functions

- double ** [allocate_2d_array](#) (int rows, int cols)
Allocates a 2D array of doubles.
- void [free_2d_array](#) (double **array, int rows)
Frees a 2D array from memory.
- int [read_natoms](#) (const char *filename)
Reads the number of atoms from an input file.
- void [read_coords_and_masses](#) (const char *filename, double **coords, double *masses, int n_atoms)
Reads atomic coordinates and masses from an input file.
- int [validate_atoms](#) (double *masses, double *epsilon, double *sigma, int n_atoms)
Validates that all atoms in the system are argon.
- FILE * [open_output](#) (const char *filename)
Opens a file for writing output.

- void `print_output` (FILE *trajectory_file, FILE *energy_file, FILE *extended_file, FILE *acceleration_file, int n_atoms, int step, double kinetic_energy, double potential_energy, double total_energy, double **coords, double **velocities, double **accelerations)
Prints simulation output.
- void `calculate_distances` (double **coords, double **distances, int n_atoms)
Calculates distances between all pairs of atoms.
- double `calculate_potential_energy` (double **distances, int n_atoms, double epsilon, double sigma)
Calculates total potential energy of the system.
- double `calculate_kinetic_energy` (double **velocities, double *masses, int n_atoms)
Calculates total kinetic energy of the system.
- double `calculate_total_energy` (double kinetic_energy, double potential_energy)
Calculates total energy of the system.
- void `check_energy` (double previous_energy, double total_energy, int step)
Checks energy conservation between simulation steps.
- void `calculate_accelerations` (double **coords, double *masses, int n_atoms, double epsilon, double sigma, double **distances, double **accelerations)
Calculates acceleration vectors for all atoms.
- void `update_positions` (double **coords, double **velocities, double **accelerations, double dt, int n_atoms)
Updates atomic positions using Verlet algorithm.
- void `update_velocities` (double **velocities, double **accelerations, double dt, int n_atoms)
Updates atomic velocities using Verlet algorithm.

2.1.1 Function Documentation

2.1.1.1 `allocate_2d_array()`

```
double ** allocate_2d_array (
    int rows,
    int cols )
```

Allocates a 2D array of doubles.

Parameters

<i>rows</i>	Number of rows in the array
<i>cols</i>	Number of columns in the array

Returns

Pointer to the allocated 2D array

Exceptions

<i>Exits</i>	with code 1 if memory allocation fails
--------------	----------------------------------------

2.1.1.2 `calculate_accelerations()`

```
void calculate_accelerations (
```



```
double ** coords,  
double * masses,  
int n_atoms,  
double epsilon,  
double sigma,  
double ** distances,  
double ** accelerations )
```

Calculates acceleration vectors for all atoms.

Parameters

<i>coords</i>	Array of atomic coordinates
<i>masses</i>	Array of atomic masses
<i>n_atoms</i>	Number of atoms
<i>epsilon</i>	Epsilon parameter for LJ potential
<i>sigma</i>	Sigma parameter for LJ potential
<i>distances</i>	2D array of pairwise distances
<i>accelerations</i>	Array to store calculated accelerations

< Variable used by [calculate_accelerations\(\)](#) for the distance

< Variable used by [calculate_accelerations\(\)](#) for the scaled potential U

< Variable used by [calculate_accelerations\(\)](#) for the distance in x

< Variable used by [calculate_accelerations\(\)](#) for the distance in y

< Variable used by [calculate_accelerations\(\)](#) for the distance in z

2.1.1.3 calculate_distances()

```
void calculate_distances (  
    double ** coords,  
    double ** distances,  
    int n_atoms )
```

Calculates distances between all pairs of atoms.

Parameters

<i>coords</i>	Array of atomic coordinates
<i>distances</i>	2D array to store pairwise distances
<i>n_atoms</i>	Number of atoms

< Variable used by [calculate_distances\(\)](#) for the distance in x

< Variable used by [calculate_distances\(\)](#) for the distance in y

< Variable used by [calculate_distances\(\)](#) for the distance in z

< Variable used by [calculate_distances\(\)](#) for the distance in 3D

2.1.1.4 calculate_kinetic_energy()

```
double calculate_kinetic_energy (
    double ** velocities,
    double * masses,
    int n_atoms )
```

Calculates total kinetic energy of the system.

Parameters

<i>velocities</i>	Array of atomic velocities
<i>masses</i>	Array of atomic masses
<i>n_atoms</i>	Number of atoms

Returns

Total kinetic energy of the system

< Variable used by [calculate_kinetic_energy\(\)](#) for updating the kinetic energy

< Varibale used by [calculate_kinetic_energy\(\)](#) for the sum of velocity squares

2.1.1.5 calculate_potential_energy()

```
double calculate_potential_energy (
    double ** distances,
    int n_atoms,
    double epsilon,
    double sigma )
```

Calculates total potential energy of the system.

Parameters

<i>distances</i>	2D array of pairwise distances
<i>n_atoms</i>	Number of atoms
<i>epsilon</i>	Epsilon parameter for LJ potential
<i>sigma</i>	Sigma parameter for LJ potential

Returns

Total potential energy of the system

< Varibale used by [calculate_potential_energy\(\)](#) for updating the potential energy

2.1.1.6 calculate_total_energy()

```
double calculate_total_energy (
    double kinetic_energy,
    double potential_energy )
```

Calculates total energy of the system.

Parameters

<i>kinetic_energy</i>	Current kinetic energy
<i>potential_energy</i>	Current potential energy

Returns

Total energy of the system

2.1.1.7 check_energy()

```
void check_energy (
    double previous_energy,
    double total_energy,
    int step )
```

Checks energy conservation between simulation steps.

Parameters

<i>previous_energy</i>	Energy from previous step
<i>total_energy</i>	Current total energy
<i>step</i>	Current simulation step

Warning

Prints warning if energy varies by more than 10%

< Variable used by [check_energy\(\)](#) for the difference in total energy between subsequent steps

2.1.1.8 free_2d_array()

```
void free_2d_array (
    double ** array,
    int rows )
```

Frees a 2D array from memory.

Parameters

<i>array</i>	Pointer to the 2D array to be freed
<i>rows</i>	Number of rows in the array

2.1.1.9 open_output()

```
FILE * open_output (
    const char * filename )
```

Opens a file for writing output.

Parameters

<i>filename</i>	Name of the output file
-----------------	-------------------------

Returns

Pointer to the opened file

Exceptions

<i>Exits</i>	with code 1 if file cannot be opened
--------------	--------------------------------------

2.1.1.10 print_output()

```
void print_output (
    FILE * trajectory_file,
    FILE * energy_file,
    FILE * extended_file,
    FILE * acceleration_file,
    int n_atoms,
    int step,
    double kinetic_energy,
    double potential_energy,
    double total_energy,
    double ** coords,
    double ** velocities,
    double ** accelerations )
```

Prints simulation output.

Parameters

<i>trajectory_file</i>	File pointer for trajectory output
<i>energy_file</i>	File pointer for energy output
<i>extended_file</i>	File pointer for extended information
<i>acceleration_file</i>	File pointer for acceleration data
<i>n_atoms</i>	Number of atoms
<i>step</i>	Current simulation step
<i>kinetic_energy</i>	Current kinetic energy
<i>potential_energy</i>	Current potential energy
<i>total_energy</i>	Current total energy
<i>coords</i>	Array of atomic coordinates
<i>velocities</i>	Array of atomic velocities
<i>accelerations</i>	Array of atomic accelerations

2.1.1.11 read_coords_and_masses()

```
void read_coords_and_masses (
    const char * filename,
    double ** coords,
    double * masses,
    int n_atoms )
```

Reads atomic coordinates and masses from an input file.

Parameters

<i>filename</i>	Name of the input file
<i>coords</i>	2D array to store coordinates
<i>masses</i>	Array to store atomic masses
<i>n_atoms</i>	Number of atoms

Exceptions

<i>Exits</i>	with code 1 if file cannot be opened
--------------	--------------------------------------

< Dummy variable used by read_coords_and_masses for reading the input

2.1.1.12 read_natoms()

```
int read_natoms (
    const char * filename )
```

Reads the number of atoms from an input file.

Parameters

<i>filename</i>	Name of the input file
-----------------	------------------------

Returns

Number of atoms

Exceptions

<i>Exits</i>	with code 1 if file cannot be opened
--------------	--------------------------------------

2.1.1.13 update_positions()

```
void update_positions (
    double ** coords,
    double ** velocities,
```

```
double ** accelerations,
double dt,
int n_atoms )
```

Updates atomic positions using Verlet algorithm.

Parameters

<i>coords</i>	Array of atomic coordinates
<i>velocities</i>	Array of atomic velocities
<i>accelerations</i>	Array of atomic accelerations
<i>dt</i>	Time step
<i>n_atoms</i>	Number of atoms

< Variable used by `update_positions` for the square of `dt`

2.1.1.14 `update_velocities()`

```
void update_velocities (
    double ** velocities,
    double ** accelerations,
    double dt,
    int n_atoms )
```

Updates atomic velocities using Verlet algorithm.

Parameters

<i>velocities</i>	Array of atomic velocities
<i>accelerations</i>	Array of atomic accelerations
<i>dt</i>	Time step
<i>n_atoms</i>	Number of atoms

2.1.1.15 `validate_atoms()`

```
int validate_atoms (
    double * masses,
    double * epsilon,
    double * sigma,
    int n_atoms )
```

Validates that all atoms in the system are argon.

Parameters

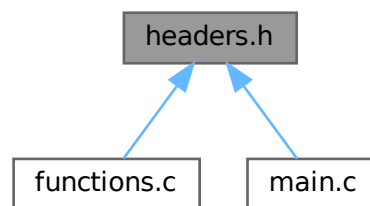
<i>masses</i>	Array of atomic masses
<i>epsilon</i>	Pointer to store the epsilon parameter
<i>sigma</i>	Pointer to store the sigma parameter
<i>n_atoms</i>	Number of atoms

Returns

1 if all atoms are valid, 0 otherwise

2.2 headers.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- `#define` [CONSTANTS_H](#)
- `#define` [ARGON_MASS](#) 39.948
- `#define` [ARGON_EPSILON](#) 0.0661
- `#define` [ARGON_SIGMA](#) 0.3345

Functions

- `double **` [allocate_2d_array](#) (int rows, int cols)
Allocates a 2D array of doubles.
- `void` [free_2d_array](#) (double **array, int rows)
Frees a 2D array from memory.
- `int` [read_natoms](#) (const char *filename)
Reads the number of atoms from an input file.
- `void` [read_coords_and_masses](#) (const char *filename, double **coords, double *masses, int n_atoms)
Reads atomic coordinates and masses from an input file.
- `int` [validate_atoms](#) (double *masses, double *epsilon, double *sigma, int n_atoms)
Validates that all atoms in the system are argon.
- `void` [calculate_distances](#) (double **coords, double **distances, int n_atoms)
Calculates distances between all pairs of atoms.
- `double` [calculate_potential_energy](#) (double **distances, int n_atoms, double epsilon, double sigma)
Calculates total potential energy of the system.
- `double` [calculate_kinetic_energy](#) (double **velocities, double *masses, int n_atoms)
Calculates total kinetic energy of the system.
- `double` [calculate_total_energy](#) (double kinetic_energy, double potential_energy)
Calculates total energy of the system.
- `void` [check_energy](#) (double previous_energy, double total_energy, int step)

Checks energy conservation between simulation steps.

- void [calculate_accelerations](#) (double **coords, double *masses, int n_atoms, double epsilon, double sigma, double **distances, double **accelerations)

Calculates acceleration vectors for all atoms.

- void [update_positions](#) (double **coords, double **velocities, double **accelerations, double dt, int n_atoms)

Updates atomic positions using Verlet algorithm.

- void [update_velocities](#) (double **velocities, double **accelerations, double dt, int n_atoms)

Updates atomic velocities using Verlet algorithm.

- FILE * [open_output](#) (const char *filename)

Opens a file for writing output.

- void [print_output](#) (FILE *trajectory_file, FILE *energy_file, FILE *extended_file, FILE *acceleration_file, int n_atoms, int step, double kinetic_energy, double potential_energy, double total_energy, double **coords, double **velocities, double **accelerations)

Prints simulation output.

2.2.1 Macro Definition Documentation

2.2.1.1 ARGON_EPSILON

```
#define ARGON_EPSILON 0.0661
```

2.2.1.2 ARGON_MASS

```
#define ARGON_MASS 39.948
```

2.2.1.3 ARGON_SIGMA

```
#define ARGON_SIGMA 0.3345
```

2.2.1.4 CONSTANTS_H

```
#define CONSTANTS_H
```

2.2.2 Function Documentation

2.2.2.1 allocate_2d_array()

```
double ** allocate_2d_array (
    int rows,
    int cols )
```

Allocates a 2D array of doubles.

Parameters

<i>rows</i>	Number of rows in the array
<i>cols</i>	Number of columns in the array

Returns

Pointer to the allocated 2D array

Exceptions

<i>Exits</i>	with code 1 if memory allocation fails
--------------	----------------------------------------

2.2.2.2 calculate_accelerations()

```
void calculate_accelerations (
    double ** coords,
    double * masses,
    int n_atoms,
    double epsilon,
    double sigma,
    double ** distances,
    double ** accelerations )
```

Calculates acceleration vectors for all atoms.

Parameters

<i>coords</i>	Array of atomic coordinates
<i>masses</i>	Array of atomic masses
<i>n_atoms</i>	Number of atoms
<i>epsilon</i>	Epsilon parameter for LJ potential
<i>sigma</i>	Sigma parameter for LJ potential
<i>distances</i>	2D array of pairwise distances
<i>accelerations</i>	Array to store calculated accelerations

< Variable used by [calculate_accelerations\(\)](#) for the distance

< Variable used by [calculate_accelerations\(\)](#) for the scaled potential U

< Variable used by [calculate_accelerations\(\)](#) for the distance in x

< Variable used by [calculate_accelerations\(\)](#) for the distance in y

< Variable used by [calculate_accelerations\(\)](#) for the distance in z

2.2.2.3 calculate_distances()

```
void calculate_distances (
    double ** coords,
    double ** distances,
    int n_atoms )
```

Calculates distances between all pairs of atoms.

Parameters

<i>coords</i>	Array of atomic coordinates
<i>distances</i>	2D array to store pairwise distances
<i>n_atoms</i>	Number of atoms

< Variable used by [calculate_distances\(\)](#) for the distance in x

< Variable used by [calculate_distances\(\)](#) for the distance in y

< Variable used by [calculate_distances\(\)](#) for the distance in z

< Variable used by [calculate_distances\(\)](#) for the distance in 3D

2.2.2.4 calculate_kinetic_energy()

```
double calculate_kinetic_energy (
    double ** velocities,
    double * masses,
    int n_atoms )
```

Calculates total kinetic energy of the system.

Parameters

<i>velocities</i>	Array of atomic velocities
<i>masses</i>	Array of atomic masses
<i>n_atoms</i>	Number of atoms

Returns

Total kinetic energy of the system

< Variable used by [calculate_kinetic_energy\(\)](#) for updating the kinetic energy

< Variable used by [calculate_kinetic_energy\(\)](#) for the sum of velocity squares

2.2.2.5 calculate_potential_energy()

```
double calculate_potential_energy (
    double ** distances,
    int n_atoms,
    double epsilon,
    double sigma )
```

Calculates total potential energy of the system.

Parameters

<i>distances</i>	2D array of pairwise distances
<i>n_atoms</i>	Number of atoms
<i>epsilon</i>	Epsilon parameter for LJ potential
<i>sigma</i>	Sigma parameter for LJ potential

Returns

Total potential energy of the system

< Variable used by [calculate_potential_energy\(\)](#) for updating the potential energy

2.2.2.6 calculate_total_energy()

```
double calculate_total_energy (
    double kinetic_energy,
    double potential_energy )
```

Calculates total energy of the system.

Parameters

<i>kinetic_energy</i>	Current kinetic energy
<i>potential_energy</i>	Current potential energy

Returns

Total energy of the system

2.2.2.7 check_energy()

```
void check_energy (
    double previous_energy,
    double total_energy,
    int step )
```

Checks energy conservation between simulation steps.

Parameters

<i>previous_energy</i>	Energy from previous step
<i>total_energy</i>	Current total energy
<i>step</i>	Current simulation step

Warning

Prints warning if energy varies by more than 10%

< Variable used by [check_energy\(\)](#) for the difference in total energy between subsequent steps

2.2.2.8 free_2d_array()

```
void free_2d_array (
    double ** array,
    int rows )
```

Frees a 2D array from memory.

Parameters

<i>array</i>	Pointer to the 2D array to be freed
<i>rows</i>	Number of rows in the array

2.2.2.9 open_output()

```
FILE * open_output (
    const char * filename )
```

Opens a file for writing output.

Parameters

<i>filename</i>	Name of the output file
-----------------	-------------------------

Returns

Pointer to the opened file

Exceptions

<i>Exits</i>	with code 1 if file cannot be opened
--------------	--------------------------------------

2.2.2.10 print_output()

```
void print_output (
    FILE * trajectory_file,
    FILE * energy_file,
    FILE * extended_file,
    FILE * acceleration_file,
    int n_atoms,
    int step,
    double kinetic_energy,
    double potential_energy,
    double total_energy,
    double ** coords,
    double ** velocities,
    double ** accelerations )
```

Prints simulation output.

Parameters

<i>trajectory_file</i>	File pointer for trajectory output
<i>energy_file</i>	File pointer for energy output
<i>extended_file</i>	File pointer for extended information
<i>acceleration_file</i>	File pointer for acceleration data
<i>n_atoms</i>	Number of atoms

Parameters

<i>step</i>	Current simulation step
<i>kinetic_energy</i>	Current kinetic energy
<i>potential_energy</i>	Current potential energy
<i>total_energy</i>	Current total energy
<i>coords</i>	Array of atomic coordinates
<i>velocities</i>	Array of atomic velocities
<i>accelerations</i>	Array of atomic accelerations

2.2.2.11 read_coords_and_masses()

```
void read_coords_and_masses (
    const char * filename,
    double ** coords,
    double * masses,
    int n_atoms )
```

Reads atomic coordinates and masses from an input file.

Parameters

<i>filename</i>	Name of the input file
<i>coords</i>	2D array to store coordinates
<i>masses</i>	Array to store atomic masses
<i>n_atoms</i>	Number of atoms

Exceptions

<i>Exits</i>	with code 1 if file cannot be opened
--------------	--------------------------------------

< Dummy variable used by read_coords_and_masses for reading the input

2.2.2.12 read_natoms()

```
int read_natoms (
    const char * filename )
```

Reads the number of atoms from an input file.

Parameters

<i>filename</i>	Name of the input file
-----------------	------------------------

Returns

Number of atoms

Exceptions

<i>Exits</i>	with code 1 if file cannot be opened
--------------	--------------------------------------

2.2.2.13 update_positions()

```
void update_positions (
    double ** coords,
    double ** velocities,
    double ** accelerations,
    double dt,
    int n_atoms )
```

Updates atomic positions using Verlet algorithm.

Parameters

<i>coords</i>	Array of atomic coordinates
<i>velocities</i>	Array of atomic velocities
<i>accelerations</i>	Array of atomic accelerations
<i>dt</i>	Time step
<i>n_atoms</i>	Number of atoms

< Variable used by update_positions for the square of dt

2.2.2.14 update_velocities()

```
void update_velocities (
    double ** velocities,
    double ** accelerations,
    double dt,
    int n_atoms )
```

Updates atomic velocities using Verlet algorithm.

Parameters

<i>velocities</i>	Array of atomic velocities
<i>accelerations</i>	Array of atomic accelerations
<i>dt</i>	Time step
<i>n_atoms</i>	Number of atoms

2.2.2.15 validate_atoms()

```
int validate_atoms (
    double * masses,
    double * epsilon,
```

```
double * sigma,
int n_atoms )
```

Validates that all atoms in the system are argon.

Parameters

<i>masses</i>	Array of atomic masses
<i>epsilon</i>	Pointer to store the epsilon parameter
<i>sigma</i>	Pointer to store the sigma parameter
<i>n_atoms</i>	Number of atoms

Returns

1 if all atoms are valid, 0 otherwise

2.3 headers.h

[Go to the documentation of this file.](#)

```
00001 // Function prototypes
00002 #ifndef FUNCTIONS_H
00003 #define FUNCTIONS_H
00004
00005 double** allocate_2d_array(int rows, int cols);
00006 void free_2d_array(double** array, int rows);
00007 int read_natoms(const char* filename);
00008 void read_coords_and_masses(const char* filename, double** coords, double* masses, int n_atoms);
00009 int validate_atoms(double* masses, double* epsilon, double* sigma, int n_atoms);
00010 void calculate_distances(double** coords, double** distances, int n_atoms);
00011 double calculate_potential_energy(double** distances, int n_atoms, double epsilon, double sigma);
00012 double calculate_kinetic_energy(double** velocities, double* masses, int n_atoms);
00013 double calculate_total_energy(double kinetic_energy, double potential_energy);
00014 void check_energy(double previous_energy, double total_energy, int step);
00015 void calculate_accelerations(double** coords, double* masses, int n_atoms, double epsilon, double
sigma, double** distances, double** accelerations);
00016 void update_positions(double** coords, double** velocities, double** accelerations, double dt, int
n_atoms);
00017 void update_velocities(double** velocities, double** accelerations, double dt, int n_atoms);
00018 FILE* open_output(const char* filename);
00019 void print_output(FILE* trajectory_file, FILE* energy_file, FILE* extended_file, FILE*
acceleration_file, int n_atoms, int step, double kinetic_energy, double potential_energy, double
total_energy, double** coords, double** velocities, double** accelerations);
00020 #endif
00021
00022 // Constants
00023 #ifndef CONSTANTS_H
00024 #define CONSTANTS_H
00025
00026 #define ARGON_MASS 39.948
00027 #define ARGON_EPSILON 0.0661 // J/mol
00028 #define ARGON_SIGMA 0.3345 // nm
00029
00030 #endif
```

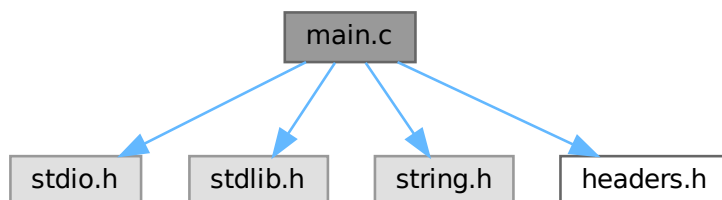
2.4 main.c File Reference

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



```
#include "headers.h"
```

Include dependency graph for main.c:



Functions

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

2.4.1 Function Documentation

2.4.1.1 main()

```
int main (  
    int argc,  
    char * argv[] )
```

< Name of the input file

< Number of atoms

< 2D array of coordinates consisting of x, y, z for each atom

< 2D array of distances for each pair of atoms

< Array of masses of each atom

< Epsilon parameter for the Lennard-Jones potential in j/mol

< Sigma parameter for the Lennard-Jones potential in nm

< 2D array of velocities consisting of vx, vy, vz for each atom

< 2D array of accelerations consisting of ax, ay, az for each atom

< Name of the file where the trajectory output is written

< File where the trajectory output is written

< Name of the file where the energies are written

< File where the energies are written

- < Name of the file where the extended trajectory with velocities is written
- < File where the extended trajectory with velocities is written
- < Name of the file where the accelerations are written
- < File where the accelerations are written
- < Number of simulation steps
- < Time step in ???
- < Variable for storing the kinetic energy
- < Variable for storing the potential energy
- < Variable for storing the total energy
- < Variable for storing the total energy of the previous step

Index

- allocate_2d_array
 - functions.c, [4](#)
 - headers.h, [12](#)
- ARGON_EPSILON
 - headers.h, [12](#)
- ARGON_MASS
 - headers.h, [12](#)
- ARGON_SIGMA
 - headers.h, [12](#)
- calculate_accelerations
 - functions.c, [4](#)
 - headers.h, [13](#)
- calculate_distances
 - functions.c, [5](#)
 - headers.h, [13](#)
- calculate_kinetic_energy
 - functions.c, [5](#)
 - headers.h, [14](#)
- calculate_potential_energy
 - functions.c, [6](#)
 - headers.h, [14](#)
- calculate_total_energy
 - functions.c, [6](#)
 - headers.h, [15](#)
- check_energy
 - functions.c, [7](#)
 - headers.h, [15](#)
- CONSTANTS_H
 - headers.h, [12](#)
- free_2d_array
 - functions.c, [7](#)
 - headers.h, [15](#)
- functions.c, [3](#)
 - allocate_2d_array, [4](#)
 - calculate_accelerations, [4](#)
 - calculate_distances, [5](#)
 - calculate_kinetic_energy, [5](#)
 - calculate_potential_energy, [6](#)
 - calculate_total_energy, [6](#)
 - check_energy, [7](#)
 - free_2d_array, [7](#)
 - open_output, [7](#)
 - print_output, [8](#)
 - read_coords_and_masses, [9](#)
 - read_natoms, [9](#)
 - update_positions, [9](#)
 - update_velocities, [10](#)
 - validate_atoms, [10](#)
- headers.h, [11](#)
 - allocate_2d_array, [12](#)
 - ARGON_EPSILON, [12](#)
 - ARGON_MASS, [12](#)
 - ARGON_SIGMA, [12](#)
 - calculate_accelerations, [13](#)
 - calculate_distances, [13](#)
 - calculate_kinetic_energy, [14](#)
 - calculate_potential_energy, [14](#)
 - calculate_total_energy, [15](#)
 - check_energy, [15](#)
 - CONSTANTS_H, [12](#)
 - free_2d_array, [15](#)
 - open_output, [17](#)
 - print_output, [17](#)
 - read_coords_and_masses, [18](#)
 - read_natoms, [18](#)
 - update_positions, [19](#)
 - update_velocities, [19](#)
 - validate_atoms, [19](#)
- main
 - main.c, [21](#)
- main.c, [20](#)
 - main, [21](#)
- open_output
 - functions.c, [7](#)
 - headers.h, [17](#)
- print_output
 - functions.c, [8](#)
 - headers.h, [17](#)
- read_coords_and_masses
 - functions.c, [9](#)
 - headers.h, [18](#)
- read_natoms
 - functions.c, [9](#)
 - headers.h, [18](#)
- update_positions
 - functions.c, [9](#)
 - headers.h, [19](#)
- update_velocities
 - functions.c, [10](#)
 - headers.h, [19](#)
- validate_atoms
 - functions.c, [10](#)
 - headers.h, [19](#)