
DOCUMENTATION AND USER GUIDE

Molecular Dynamics Program for Argon

Pauline Schütt & Albert Makhmudov

Version 1.0

Updated on: 10/01/2025

1 The program

This project contains a program for running molecular dynamics (MD) simulations for Argon. The program reads atomic coordinates and masses from an input file and performs a MD simulation using the Verlet algorithm and the Lennard-Jones potential. The trajectory in XYZ format, as well as an extended trajectory file including velocities, the accelerations and energies for each step are generated as output.

2 Installation

2.1 Prerequisites

Ensure you have the following installed on your system.

```
gcc
make
```

2.2 Installation

1. Clone the repository:

```
git clone https://github.com/pauline-schtt/tccm-homeworks/tree/master/project3
cd project3
```

2. Compile the program using `make`:

```
make
```

3. Then, run the program:

```
./MD <path_to_the_input_file>
```

Installation instructions can also be found in the `INSTALL.md` file.

3 Usage

To run the molecular dynamics simulation, provide the full path to the input file containing the atomic coordinates and masses as an argument to the program. An example input file can be found in `data/inp.txt`.

Example:

```
./MD data/inp.txt
```

4 Input format

Atomic coordinates are provided in nm in the following format:

```
<number of atoms>
<x_1>   <y_1>   <z_1>   <mass_1>
<x_2>   <y_2>   <z_2>   <mass_2>
...
<x_n>   <y_n>   <z_n>   <mass_n>
```

Example:

```
3
0.0   0.0   0.0   39.948
0.0   0.0   0.5   39.948
0.1   0.2  -0.5   39.948
```

An example input file can be found in the `data` folder.

5 Output format

The program produces the following output files:

- **trajectory.xyz**: atomic coordinates for each step of the simulation in XYZ format, kinetic, potential and total energy are provided on the comment line
- **trajectory_velocities.xyz**: similar to trajectory.xyz, but additionally contains the velocities for each atom
- **energies**: kinetic, potential and total energy for each step of the simulation
- **accelerations**: acceleration in x, y and z direction for each atom for each step of the simulation

The trajectory in XYZ format can be opened and visualized, e.g. with the VMD or JMOL software.

Example output files can be found in the `tests` folder.

6 Default values

The program uses the following default values:

- number of steps: 1000
- time step: 0.2
- mass of Argon: 39.948 g/mol
- ϵ : 0.661 j/mol
- σ : 0.3345 nm

Velocities are initialized to zero.

No thermostat is used, but conversation of the total energy is checked at each step and the user is warned if the energy varies significantly.

7 Tests

The program was tested on MacOS Sequoia 15.2 and Ubuntu 24.04. You can find an example input file in the `data` folder, while the corresponding output files can be found in the `test` folder. As no random velocity initialization is performed, you should obtain qualitatively similar results when running the program with the test input.

8 Cleaning up

To clean up the compiled files, run:

```
make clean
```

9 Troubleshooting

If you encounter any issues during the installation, ensure that your versions of the prerequisites meet the required versions mentioned above. In case you're still facing issues, feel free to reach out to any of the contributors or opening an issue on GitHub.

10 License

The code is licensed under the GNU GENERAL PUBLIC LICENSE.

11 Acknowledgments

This project is based on data and instructions provided by Abdallah Ammar, Yann Damour, Peter Reinhardt and Anthony Scemama, available at <https://github.com/scemama/tccm-homeworks>.

Appendix

Detailed documentation on the file contents, functions and variables in the code are provided in the documentation as generated using Doxygen (<https://www.doxygen.nl/>) in the appendix.

Molecular Dynamics Code for Argon

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 Detailed Description	4
2.1.2 Function Documentation	4
2.1.2.1 allocate_2d_array()	4
2.1.2.2 calculate_accelerations()	5
2.1.2.3 calculate_distances()	5
2.1.2.4 calculate_kinetic_energy()	6
2.1.2.5 calculate_potential_energy()	6
2.1.2.6 calculate_total_energy()	6
2.1.2.7 calculate_U()	7
2.1.2.8 check_energy()	7
2.1.2.9 free_2d_array()	8
2.1.2.10 lennard_jones_potential()	8
2.1.2.11 open_output()	8
2.1.2.12 print_output()	9
2.1.2.13 read_coords_and_masses()	9
2.1.2.14 read_natoms()	10
2.1.2.15 update_positions()	10
2.1.2.16 update_velocities()	11
2.1.2.17 validate_atoms()	11
2.2 headers.h File Reference	11
2.2.1 Detailed Description	13
2.2.2 Macro Definition Documentation	13
2.2.2.1 ARGON_EPSILON	13
2.2.2.2 ARGON_MASS	13
2.2.2.3 ARGON_SIGMA	13
2.2.2.4 CONSTANTS_H	13
2.2.3 Function Documentation	13
2.2.3.1 allocate_2d_array()	13
2.2.3.2 calculate_accelerations()	14
2.2.3.3 calculate_distances()	14
2.2.3.4 calculate_kinetic_energy()	15
2.2.3.5 calculate_potential_energy()	15
2.2.3.6 calculate_total_energy()	15
2.2.3.7 check_energy()	16
2.2.3.8 free_2d_array()	16
2.2.3.9 open_output()	16
2.2.3.10 print_output()	17

2.2.3.11 read_coords_and_masses()	17
2.2.3.12 read_natoms()	18
2.2.3.13 update_positions()	18
2.2.3.14 update_velocities()	19
2.2.3.15 validate_atoms()	19
2.3 headers.h	20
2.4 main.c File Reference	20
2.4.1 Detailed Description	21
2.4.2 Function Documentation	21
2.4.2.1 main()	21
2.4.3 Variables	21
Index	23

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

functions.c	Contains the functions associated with the molecular dynamics simulation	3
headers.h	Contains the function headers	11
main.c	Contains the main program	20

Chapter 2

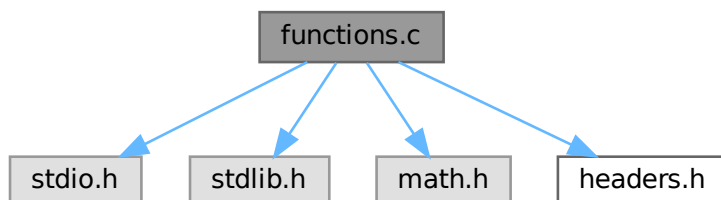
File Documentation

2.1 functions.c File Reference

Contains the functions associated with the molecular dynamics simulation.

```
#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.
- static double [lennard_jones_potential](#) (double r, double epsilon, double sigma)
Calculates the Lennard-Jones potential between two 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.
- static double [calculate_U](#) (double r, double epsilon, double sigma)
Helper function for acceleration calculation.
- 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 Detailed Description

Contains the functions associated with the molecular dynamics simulation.

2.1.2 Function Documentation

2.1.2.1 [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
--------------	--

2.1.2.2 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

2.1.2.3 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

2.1.2.4 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

2.1.2.5 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

2.1.2.6 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

2.1.2.7 calculate_U()

```
static double calculate_U (  
    double r,  
    double epsilon,  
    double sigma ) [static]
```

Helper function for acceleration calculation.

Parameters

<i>r</i>	Distance between atoms
<i>epsilon</i>	Epsilon parameter for LJ potential
<i>sigma</i>	Sigma parameter for LJ potential

Returns

U value

2.1.2.8 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%

2.1.2.9 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

2.1.2.10 lennard_jones_potential()

```
static double lennard_jones_potential (
    double r,
    double epsilon,
    double sigma ) [static]
```

Calculates the Lennard-Jones potential between two atoms.

Parameters

<i>r</i>	Distance between atoms
<i>epsilon</i>	Epsilon parameter for LJ potential
<i>sigma</i>	Sigma parameter for LJ potential

Returns

Value of the Lennard-Jones potential

2.1.2.11 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
--------------	--------------------------------------

2.1.2.12 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

2.1.2.13 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
-----------------	------------------------

Parameters

<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
--------------	--------------------------------------

2.1.2.14 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
--------------	--------------------------------------

2.1.2.15 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

2.1.2.16 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

2.1.2.17 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

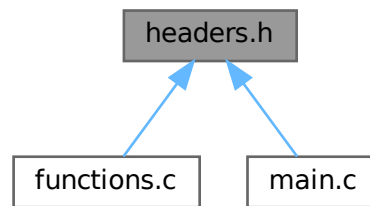
Returns

1 if all atoms are valid, 0 otherwise

2.2 headers.h File Reference

Contains the function headers.

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 Detailed Description

Contains the function headers.

2.2.2 Macro Definition Documentation

2.2.2.1 ARGON_EPSILON

```
#define ARGON_EPSILON 0.0661
```

2.2.2.2 ARGON_MASS

```
#define ARGON_MASS 39.948
```

2.2.2.3 ARGON_SIGMA

```
#define ARGON_SIGMA 0.3345
```

2.2.2.4 CONSTANTS_H

```
#define CONSTANTS_H
```

2.2.3 Function Documentation

2.2.3.1 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
--------------	--

2.2.3.2 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

2.2.3.3 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

2.2.3.4 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

2.2.3.5 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

2.2.3.6 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

2.2.3.7 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%

2.2.3.8 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

2.2.3.9 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
--------------	--------------------------------------

2.2.3.10 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

2.2.3.11 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
--------------	--------------------------------------

2.2.3.12 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
--------------	--------------------------------------

2.2.3.13 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

2.2.3.14 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

2.2.3.15 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
00006 #ifndef FUNCTIONS_H
00007 #define FUNCTIONS_H
00008
00009 double** allocate_2d_array(int rows, int cols);
00010 void free_2d_array(double** array, int rows);
00011 int read_natoms(const char* filename);
00012 void read_coords_and_masses(const char* filename, double** coords, double* masses, int n_atoms);
00013 int validate_atoms(double* masses, double* epsilon, double* sigma, int n_atoms);
00014 void calculate_distances(double** coords, double** distances, int n_atoms);
00015 double calculate_potential_energy(double** distances, int n_atoms, double epsilon, double sigma);
00016 double calculate_kinetic_energy(double** velocities, double* masses, int n_atoms);
00017 double calculate_total_energy(double kinetic_energy, double potential_energy);
00018 void check_energy(double previous_energy, double total_energy, int step);
00019 void calculate_accelerations(double** coords, double* masses, int n_atoms, double epsilon, double
    sigma, double** distances, double** accelerations);
00020 void update_positions(double** coords, double** velocities, double** accelerations, double dt, int
    n_atoms);
00021 void update_velocities(double** velocities, double** accelerations, double dt, int n_atoms);
00022 FILE* open_output(const char* filename);
00023 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);
00024 #endif
00025
00026 // Constants
00027 #ifndef CONSTANTS_H
00028 #define CONSTANTS_H
00029
00030 #define ARGON_MASS 39.948
00031 #define ARGON_EPSILON 0.0661 // J/mol
00032 #define ARGON_SIGMA 0.3345 // nm
00033
00034 #endif

```

2.4 main.c File Reference

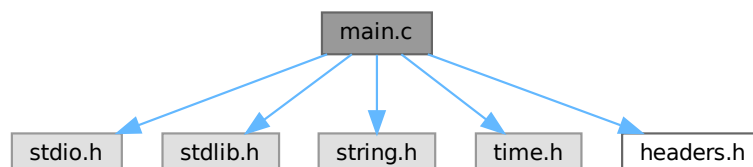
Contains the main program.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "headers.h"

```

Include dependency graph for main.c:



Functions

- `int main (int argc, char *argv[])`
The main entry point of the program.

2.4.1 Detailed Description

Contains the main program.

2.4.2 Function Documentation

2.4.2.1 main()

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

The main entry point of the program.

This program manages reading of the input, performing the MD simulation and generation of output.

Returns

int Returns 0 upon successful execution.

2.4.3 Variables

- `const char* filename`: Name of the input file.
- `int n_atoms`: Number of atoms.
- `double** coords`: 2D array of coordinates consisting of x, y, z for each atom.
- `double** distances`: 2D array of distances for each pair of atoms.
- `double* masses`: Array of masses of each atom.
- `double epsilon`: Epsilon parameter for the Lennard-Jones potential in j/mol.
- `double sigma`: Sigma parameter for the Lennard-Jones potential in nm.
- `double** velocities`: 2D array of velocities consisting of vx, vy, vz for each atom.
- `double** accelerations`: 2D array of accelerations consisting of ax, ay, az for each atom.
- `const char* trajectory_name`: Name of the file where the trajectory output is written.
- `FILE* trajectory_file`: File where the trajectory output is written.
- `const char* energy_name`: Name of the file where the energies are written.
- `FILE* energy_file`: File where the energies are written.
- `const char* extended_name`: Name of the file where the extended trajectory with velocities is written.
- `FILE* extended_file`: File where the extended trajectory with velocities is written.

- `const char* acceleration_name`: Name of the file where the accelerations are written.
- `FILE* acceleration_file`: File where the accelerations are written.
- `int n_steps`: Number of simulation steps.
- `double dt`: Time step.
- `double kinetic_energy`: Variable for storing the kinetic energy.
- `double potential_energy`: Variable for storing the potential energy.
- `double total_energy`: Variable for storing the total energy.
- `double previous_energy`: Variable for storing the total energy of the previous step. Name of the input file

Index

- allocate_2d_array
 - functions.c, [4](#)
 - headers.h, [13](#)
- ARGON_EPSILON
 - headers.h, [13](#)
- ARGON_MASS
 - headers.h, [13](#)
- ARGON_SIGMA
 - headers.h, [13](#)
- calculate_accelerations
 - functions.c, [5](#)
 - headers.h, [14](#)
- calculate_distances
 - functions.c, [5](#)
 - headers.h, [14](#)
- calculate_kinetic_energy
 - functions.c, [5](#)
 - headers.h, [14](#)
- calculate_potential_energy
 - functions.c, [6](#)
 - headers.h, [15](#)
- calculate_total_energy
 - functions.c, [6](#)
 - headers.h, [15](#)
- calculate_U
 - functions.c, [7](#)
- check_energy
 - functions.c, [7](#)
 - headers.h, [16](#)
- CONSTANTS_H
 - headers.h, [13](#)
- free_2d_array
 - functions.c, [7](#)
 - headers.h, [16](#)
- functions.c, [3](#)
 - allocate_2d_array, [4](#)
 - calculate_accelerations, [5](#)
 - calculate_distances, [5](#)
 - calculate_kinetic_energy, [5](#)
 - calculate_potential_energy, [6](#)
 - calculate_total_energy, [6](#)
 - calculate_U, [7](#)
 - check_energy, [7](#)
 - free_2d_array, [7](#)
 - lennard_jones_potential, [8](#)
 - open_output, [8](#)
 - print_output, [9](#)
 - read_coords_and_masses, [9](#)
 - read_natoms, [10](#)
 - update_positions, [10](#)
 - update_velocities, [11](#)
 - validate_atoms, [11](#)
- headers.h, [11](#)
 - allocate_2d_array, [13](#)
 - ARGON_EPSILON, [13](#)
 - ARGON_MASS, [13](#)
 - ARGON_SIGMA, [13](#)
 - calculate_accelerations, [14](#)
 - calculate_distances, [14](#)
 - calculate_kinetic_energy, [14](#)
 - calculate_potential_energy, [15](#)
 - calculate_total_energy, [15](#)
 - check_energy, [16](#)
 - CONSTANTS_H, [13](#)
 - free_2d_array, [16](#)
 - open_output, [16](#)
 - print_output, [17](#)
 - read_coords_and_masses, [17](#)
 - read_natoms, [18](#)
 - update_positions, [18](#)
 - update_velocities, [19](#)
 - validate_atoms, [19](#)
- lennard_jones_potential
 - functions.c, [8](#)
- main
 - main.c, [21](#)
- main.c, [20](#)
 - main, [21](#)
- open_output
 - functions.c, [8](#)
 - headers.h, [16](#)
- print_output
 - functions.c, [9](#)
 - headers.h, [17](#)
- read_coords_and_masses
 - functions.c, [9](#)
 - headers.h, [17](#)
- read_natoms
 - functions.c, [10](#)
 - headers.h, [18](#)
- update_positions
 - functions.c, [10](#)

headers.h, [18](#)
update_velocities
functions.c, [11](#)
headers.h, [19](#)

validate_atoms
functions.c, [11](#)
headers.h, [19](#)