Molecular mechanics

Author: Martijn Oele

GitHub: Martijn-075

VUID:

Table of Contents

[1. Modules 3](#_Toc130483957)

[1.1. Molecule module 3](#_Toc130483958)

[1.2. Energy module 3](#_Toc130483959)

[1.3. Metropolis module 3](#_Toc130483960)

[1.4. Math module 3](#_Toc130483961)

[1.5. Constants module 3](#_Toc130483962)

[2. Data types 2](#_Toc130483963)

[2.1. Atom 2](#_Toc130483964)

[2.2. Bond 2](#_Toc130483965)

[2.3. Bonds\_angle 2](#_Toc130483966)

[2.4. Torsion 2](#_Toc130483967)

[2.5. Molecule 3](#_Toc130483968)

[3. Subroutines and functions 4](#_Toc130483969)

[3.1. Molecule subroutines 4](#_Toc130483970)

[3.1.1. Read atom(mol, filename) and Write atom(mol, filename) 4](#_Toc130483971)

[3.1.2. Bond angle(mol) 4](#_Toc130483972)

[3.1.3. Angle bonds(mol) 4](#_Toc130483973)

[3.1.4. Angle torsion(mol) 4](#_Toc130483974)

[3.1.5. Create molecule(mol) 4](#_Toc130483975)

[3.1.6. Delete molecule(mol) 4](#_Toc130483976)

[3.2. Energy functions 4](#_Toc130483977)

[3.2.1. Stretch energy(mol) 4](#_Toc130483978)

[3.2.2. Bending energy(mol) 5](#_Toc130483979)

[3.2.3. Torsion energy(mol) 5](#_Toc130483980)

[3.2.4. Electrostatic energy(mol) 5](#_Toc130483981)

[3.2.5. Van der Waals energy(mol) 5](#_Toc130483982)

[3.2.6. Forcefield energy(mol) 5](#_Toc130483983)

[3.3. Metropolis subroutines 6](#_Toc130483984)

[3.3.1. Random atom metropolis(mol) 6](#_Toc130483985)

[3.3.2. Metropolis(mol) 6](#_Toc130483986)

[3.3.3. Minimize energy 6](#_Toc130483987)

[3.4. Supporting math functions 6](#_Toc130483988)

[3.4.1. Deg to rad(deg) and rad to deg(rad) 6](#_Toc130483989)

[3.4.2. Cross product(vec1, vec2) 6](#_Toc130483990)

[3.4.3. Vector length(vector) 6](#_Toc130483991)

# Modules

## Molecule module

The molecule module contains the data types used to simplify argument passing in future subroutines and functions. The main data type molecule is split into several data types such as bond, bond angle and torsion. This module also contains the subroutines associated with calculating the relevant parameters of the molecule such as the bonds, bonds angles and torsion angles

## Energy module

The energy module contains alle the function for calculating the different energies of the molecule. This includes the stretch, bending, torsion, electrostatic and van der Waals energies. All these energies are combined into the forcefield energy.

## Metropolis module

The metropolis module contains the metropolis algorithm which minimizes the energy of the molecule by randomly sampling and changing the atoms position within a defined radius and checks if the new forcefield energy is lower then the previous iteration. If the energy is higher there is still a nonzero probability that the new atoms position are accepted based on the Boltzmann distribution. This is repeated until the new atoms positions are rejected a certain amount of time in a row. The energy of the molecule is then believed to be in a (local) minimum.

## Math module

Not all basic math function used in this program were available as Fortran functions so these simple functions were implemented. This includes degree and radian conversion, 3D cross product and vector length functions.

## Constants module

The constants module contains all constant that are used throughout the program. It contains the number of bytes for all real numbers and constants for the molecule, energy and metropolis module.

# Data types

For real number realkind = 8 by default.

## Atom

|  |  |
| --- | --- |
| Data type | Description |
| character(1) :: element | The element of the atom (in this project can only be C for carbon or H for hydrogen). |
| real(realkind) :: cords(3) | The 3D cartesian coordinates of the atom. |

## Bond

|  |  |
| --- | --- |
| Data type | Description |
| integer :: link(2) | The atom indices of the atoms that are forming a bond. |
| character(2) :: type | The type of bond (in this project can only be CC for carbon carbon bond of CH for carbon hydrogen bond. |
| real(realkind) :: length | The length of the bond extracted from the distance array (see 2.5. Molecule). |
| real(realkind) :: vector(3) | The 3D vector of the bond calculated using the boding atoms coordinates. |

## Bonds\_angle

|  |  |
| --- | --- |
| Data type | Description |
| type (bond) :: bonds(2) | The 2 connecting bonds that form a angle.  See 2.2. Bonds. |
| real(realkind) :: angle | The angle of the bonds in degrees. |

## Torsion

|  |  |
| --- | --- |
| Data type | Description |
| type (bond) :: end\_bonds(2) | The 2 end bonds that have an angle in the Newman projection.  See 2.2. Bond. |
| type (bond) :: center\_bond | The central (CC) bond that connects the end bonds.  See 2.2. Bond. |
| real(realkind) :: angle | The angle of the two end bond from Newman projection. |

## Molecule

|  |  |
| --- | --- |
| Data type | Description |
| type (atom), allocatable :: atoms(:) | The atoms of the molecule.  See 2.1. atom. |
| type (bond), allocatable :: bonds(:) | The bonds of the molecule.  See 2.2. Bonds. |
| type (bond\_angle), allocatable :: angles(:) | The angles between 2 connecting bonds in the molecule.  See 2.3. Bonds\_angle. |
| type (torsion), allocatable :: torsion\_angles(:) | The torsion angle between 3 connecting bonds. The molecule must contain at least 1 CC bond otherwise not used.  See 2.4.Torsion. |
| real(realkind), allocatable :: distance(:,:) | Array of all the distances between the atoms in the molecules. Is used to determine if atoms are boding and for nonbonding interactions. |
| logical, allocatable :: bonding(:,:) | Supports the distance array and stores if atoms are bonding. |

# Subroutines and functions

## Molecule subroutines

### Read atom(mol, filename) and Write atom(mol, filename)

Parameters:  
Atoms is a derived atom type (type (bond)) see 2.1. atom. (In practice the sub type atoms from the derived type molecule is passed in this function (molecule%atoms))  
Filename is a character string containing the filename (preferably ending in .xyz) of the file to read or written to.

These subroutines read and write the coordinates and elements of the atoms.

Operation (read atom):

1. The file is opened
2. The number of atoms is used to allocate the atoms vector in the molecule type
3. For every line (number of atoms) the element (character(1)) and the x, y, z coordinates 3 (real numbers) are read in
4. The file is closed

Operation (write atom)

1. The file is opened
2. The number of atoms is written
3. A empty line is written
4. The element (character(1) and the x, y, z coordinates are written on one line per atom
5. The file is closed

The file has the following standard layout:

Number of atoms

(Empty line)

Element x coordinate y coordinate z coordinate

…

### Bonds atom(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

The bonds atom subroutine calculates the distances between all atoms (later on used for nonbonding interactions as well) and determines if two atoms are bonding based on the elements of the atoms and there distance. The bonds are stored in the bonds type (molecule%bonds).

Operation:

1. The bonds vector inside the molecule type is allocated at the same size as the atoms vector (number of atoms)
2. The distance and bonding that are part of the molecule type are allocated as a square matrix with the same size as the atoms vector (and bonds) in both dimensions
3. The distances between all atoms are calculated and stored in the distance array
4. All unique atom pairs are checked if they are bonding by checking:
   1. For carbon carbon bonds:
      1. Check if both atoms are carbon (element == C)
      2. Check if the distance is within 10% of the optimal carbon carbon single bond
   2. For carbon hydrogen:
      1. Check if the first atom is a carbon and the second a hydrogen
      2. Check if the distance is within 10% of the optimal carbon hydrogen bond
5. If one of these criteria is met:
   1. The indices of the atoms are stored in the bond%link
   2. The distance (now bond length) is stored in the bond%length
   3. The bond vector is calculated and stored in bond%vector
   4. In the bonding array the position is set to true
   5. The bond type (CC or CH) is stored in bond%type

### Angle bonds(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

### Angle torsion(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

### Create molecule(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

### Delete molecule(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

## Energy functions

### Stretch energy(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

Where A and B are atoms that are bonding, is a constant that depends on the type of bond (CC or CH), is the bond length and is the optimal bond length and depends on the type of bond.

Before the stretch energy function can be called the bonds atoms subroutine (3.1.2.) must be called in advance, otherwise this function will give an error!

Operation:

1. Based the type of bond the stretch energy of the bond is calculated
2. The stretch energy of all bonds is summed up and returned

### Bending energy(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

Where A and B are bonds that are connected with an central atom, is an constant that depends on the type bond types, is the angle in degrees between the two connected bonds and is the optimal angle of the bonds (in this project only SP3 hybridized carbons are used so the angel will be the same for all bon combinations)

Before the bending energy function can be called the angle bond subroutine (3.1.3.) must be call in advance, otherwise this function will give an error!

Operation:

1. For all connecting bonds pairs the two bond types are checked
2. The bending energy per connected bond pair is calculated
3. The energies of all bending bond pairs is summed and returned

### Torsion energy(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

Where A, B and C are 3 connecting bonds where B must be an CC bond (in this project), is a constant depended on the central bond and thus is the same for all torsion angles, n is a constant based on the possible number of torsion angles for central bond, is the torsion (dihedral) angle in degrees, is a correction of the torsion angle

Before the torsion angle energy function can be called the angle torsion subroutine (3.1.4.) must be called in advance, otherwise this function will give an error!

Operation:

1. Checked if there is a CC bond otherwise there are no torsion angle and energy
2. For all torsion angles the torsion energy is calculated
3. The torsion energies are summed and returned

### Electrostatic energy(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

Where i and j are nonbonding atom pairs, are the charges of both atoms depending on the atom element and is the coulomb constate in (Kcal/mol) \* A / e2

Before the electrostatic energy function can be called the bonds atoms subroutine (3.1.2.) must be called in advance, otherwise this function will give an error!

Operation:

1. For all unique atom pairs it is checked if they are nonbonding by checking the bonding array
2. Depending on both elements of the atoms the electrostatic energy is calculated
3. The electrostatic energies are summed and returned

### Van der Waals energy(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

Where i and j are nonbonding atom pair,

Before the van der Waals energy function can be called the bonds atoms subroutine (3.1.2.) must be called in advance, otherwise this function will give an error!

Operation:

### Forcefield energy(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

The total energy is calculated by

Function:

Before the forcefield energy function can be called the create molecule subroutine (3.1.5.) must be called in advance, otherwise the function will give an error

## Metropolis subroutines

### Random atom metropolis(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

### Metropolis(mol)

Parameters:  
Mol is a derived molecule type (type (molecule)) see 2.5. Molecule

### Minimize energy

## Supporting math functions

Some addition math functions that are used throughout the program.

### Deg to rad(deg) and rad to deg(rad)

Parameters:  
Deg and rad are real numbers

Deg to rad converts degrees to radian. Rad to deg converts radian to degrees. These functions are needed as Fortran uses radian and the used energy function use degrees.

### Cross product(vec1, vec2)

Parameters:  
Vec1 and Vec2 are 3D real number vectors

The cross product has no standard Fortran function so one for 3D vectors needed to be implemented following the function below.

Function:

A and B are 3D vectors.

### Vector length(vector)

Parameters:  
Vector is an 3D vector

To save repeating the same function a function to calculate the length of a vector was implemented using the function below (square root of the dot product).

Function:

A is an 3D vector.