Verlet Algorithm in Python

Introduction

The goal is to write a simple molecular dynamics program in Python. The program will read initial atomic coordinates, and will print in a `xyz` file the coordinates of the atoms at every step of the dynamics. The `xyz` file will be readable by `molden` for example.

We will have to program:

1. The potential energy for a pair of atoms (Lennard-Jones potential)
2. The potential and kinetic energy of a system of Natoms
3. The acceleration of the particles by finite difference
4. The Verlet algorithm

The following parameters will be used for the Argon atom:

- mass : 39.948 g/mol
- ϵ= 0.0661 j/mol
- σ= 0.3345 nm

The atom coordinates will be expressed in nm.

Lennard-Jones potential

Write a function that computes the Lennard-Jones potential:

$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

The values of $\sigma$and ϵwill be given as global variables at the beginning of the script.

This is the output you should get:

```
Enter inter-atomic distance?
0.3
V = 0.468192418088
```

Molecular geometry

Read the molecular data

Modify your program such that it reads from the standard input:

- The number of atoms
- The atomic masses
- The atomic coordinates (x,y,z on each line)

```
def main():
    Natoms = read_Natoms()
    mass  = read_atom_mass(Natoms)
    coord = read_atom_coord(Natoms)
    print mass
    print coord
```

The output should be

```
Number of atoms?
3
For each atom: mass
40
10
20
For each atom: x, y, z
0 0 0
1 2 3
-1 0 2
mass  = [40.0, 10.0, 20.0]
coord = [[0.0, 0.0, 0.0], [1.0, 2.0, 3.0], [-1.0, 0.0, 2.0]]
```

Compute the distance matrix

Add a function to your program to compute the distance matrix:

```
def main():
    Natoms = read_Natoms()
    coord = read_atom_coord(Natoms)
    distances = get_distances(coord)
    for d in distances:
       print d
```

The output should be

```
Number of atoms?
3
For each atom: x, y, z
0 0 0
1 2 3
-1 0 2
[0.0, 3.7416573867739413, 2.23606797749979]
[3.7416573867739413, 0.0, 3.0]
[2.23606797749979, 3.0, 0.0]
```

Potential for multiple atoms

Change your Lennard-Jones function such that it computes the Lennard-Jones potential of the whole system:

$$V_{\mathrm{LJ}} = \sum_{i=1}^{N_{\mathrm{atoms}}} \sum_{j>i}^{N_{\mathrm{atoms}}} V(r_{ij})$$

The main function should look like this:

```
def main():
  Natoms = read_Natoms()
  coord = read_atom_coord(Natoms)
  print V_lj(coord)
```

And the output:

```
Number of atoms?
3
For each atom: x, y, z
0 0 0
0 0 .3
.1 .2 -.3
0.396856906955
```

Total energy

Write a function that computes the total energy of the system:

$$E = T + V_{\mathrm{LJ}} \qquad T = \frac{1}{2}\sum_{i=1}^{N_{\mathrm{atoms}}} m_i v_i^2$$

Your main should look like this:

```
def main():
    Natoms = read_Natoms()
    mass = read_atom_mass(Natoms)
    coord = read_atom_coord(Natoms)
    velocity = [ [0.1,0.2,0.3] for i in range(Natoms) ]
    print E_tot(coord,mass,velocity)
```

And your output like that:

```
Number of atoms?
3
For each atom: mass
10
20
15
For each atom: x, y, z
0 0 0
0 0 .3
.1 .2 -.3
3.54685690696
```

Acceleration

The acceleration vector is given by:

$$a_{x_i} = -\frac{1}{m_i}\frac{\partial V}{\partial x_i}$$

where x_i is the $x$coordinate of atom i.

Write the function to compute the approximate derivative of the potential with respect to atomic coordinates:

$$\frac{\partial V}{\partial x_i} \sim \frac{V(x_i + \Delta x_i) - V(x_i - \Delta x_i)}{2\Delta x_i}$$

Here is the main function:

```
def main():
    Natoms = read_Natoms()
    mass = read_atom_mass(Natoms)
```

```
    coord = read_atom_coord(Natoms)
    velocity = [ [0.1,0.2,0.3] for i in range(Natoms) ]
    print get_acceleration(coord,mass)
```

and the expected output

```
f atoms?
3
For each atom: mass
10
20
15
For each atom: x, y, z
0 0 0
0 0 .3
.1 .2 -.3
[[-0.0012143469700354181, -0.0024287378274090443, -2.8852483886704636],
[0.0003772257075318475, 0.0007544514316476514, 1.4421824477393457],
[0.00030659703662931176, 0.000613223309409161, 0.0005889954611815185]]
```

Verlet Algorithm

The Verlet algorithm is the following:

$$\mathbf{r}^{n+1} = \mathbf{r}^n + \mathbf{v}^n \Delta t + \mathbf{a}^n \frac{\Delta t^2}{2}$$
$$\mathbf{v}^{n+1} = \mathbf{v}^n + \frac{1}{2}(\mathbf{a}^n + \mathbf{a}^{n+1})\Delta t$$

where:

- $\mathbf{r}$ is the position vector (atomic coordinates)
- $\mathbf{v}$ is the velocity vector
- $\mathbf{a}$ is the acceleration vector
- Δt is a time step

Write the verlet algorithm in a function. At each time step, print the atom coordinates (for the Argon atom) in `xyz` format with the total energy in the title.

Here is the main fuction:

```
def main():
  Natoms = read_Natoms()
  mass = [ mass_Argon for i in range(Natoms) ]
  coord = read_atom_coord(Natoms)
  velocity = [ [0.,0.,0.] for i in range(Natoms) ]
  coord,velocity = verlet(Nsteps=1000,Delta_t=.1,coord,mass,velocity)
```

and the ouput:

```
Number of atoms?
3
For each atom: x, y, z
0 0 0
0 0 .3
.1 .2 -.3
3
```

```
Step 0   E = 0.396856906955
Ar   0.000000   0.000000   0.000000
Ar   0.000000   0.000000   0.300000
Ar   0.100000   0.200000  -0.300000
3
Step 1   E = 0.371644652669
Ar  -0.000002  -0.000003  -0.003611
Ar   0.000001   0.000002   0.303610
Ar   0.100001   0.200001  -0.299999
[...]
3
Step 998   E = 0.377839040613
Ar  -1.656240  -3.312489  -5.790241
Ar   0.000898   0.001795  10.642537
Ar   1.755342   3.510694  -4.852296
3
Step 999   E = 0.377839040613
Ar  -1.657894  -3.315796  -5.796029
Ar   0.000898   0.001797  10.652901
Ar   1.756995   3.513999  -4.856872
```