Reproduction of A. Rahman paper Results

Project : 2

Name: Satyendra Pandey

Roll No.: 14807611

Course : Molecular Dynamics and Simulations.

1 Introduction

In this project, We are goint to write a molecular dynamics code to simulate liquid argon atoms. We are also going to reproduce the results " **Velocity autocorrelation and Radial pair distribution**" from classical paper of A. Rahman.

2 Molecular Simulation

Molecular dynamics (MD) is a computer simulation method for studying the physical movements of atoms and molecules. The atoms and molecules are allowed to interact for a fixed period of time, giving a view of the dynamic evolution of the system.

2.1 Description of project

In this project We have used 864 atoms of argon and arranged them in a cubical box with periodic box conditions as well as we make sure that two particles are atleast σ distance awake from other molecules.

2.2 Parameters Used

Here is the list of parameters used in MD simulations :

- $\sigma = 3.4$
- R-cutoff = 7.65
- num-of-particles = 864
- mass = $6.69 * 10^{-26}$ Kg
- kb = $1.38 * 10^{-23}$ J/K
- Temp = 94.4 K
- box size = 34.778 A
- Density = 1.374 gm/cm^3

2.3 Algorithms Used In code

Here is some of the key steps taken to Write a Optimised Algo for MD Simulation.

Random velocity and position Allocation :

To Generate Random velocity and position I have used **Numpy** module in python which can give use normnal distribution by providing mean and standard deviation of the distribution.
I used linespace to create a solid cubic lattice and then removed atoms from different positions.
For velocity I have given values of sigma and mean such that It gives us a distribution with temperature as much needed.

Algorithm For periodic Boundry condition :

Here is code from force and potential calculation of Periodic box condition

```
float force_cal(float posx, float posy, float posz, float * accx, float * accy, float * accz)
{
  float potential = 0;
// For all Atoms
  int flag = 0;
  if ((posx-17.39) > 9.6) flag++;
  if ((posy-17.39) > 9.6) flag++;
  if ((posz-17.39) > 9.6) flag++;

for (int i = 0; i<27; i++)
{
  // Mapping to 3D
  int num = i;
  int cor_x, cor_y, cor_z;
  cor_x = (num/9)-1;
  cor_y = ((num%9)/3)-1;
  cor_z = (num%3)-1;

if ((abs(cor_x) + abs(cor_y) + abs(cor_z)) <= flag)
{
  for (int j = 0; j<864; j++)
        {
            float r = pow( (pow( (pos[j][0] + cor_x * 34.78 )- posx, 2) + pow((pos[j][1] + cor_y * 34.78 )- posy, 2) + pow((pos[j][2] + cor_z * 34.78 )- posz,
            if (r < 7.65 && r > .1)
            {
                potential = potential + 4 * epsilon * ((pow((sigma/r),12))  - (pow((sigma/r),6)));
                float force = -4000 * kbm * epsilon * ((pow((sigma/r),13) * 12)  - (pow((sigma/r),7) * 6))/ sigma;
                *accx = *accx + force * ((pos[j][0] + cor_x * 34.78)-posx)/r;
                *accy = *accy + force * ((pos[j][1] + cor_y * 34.78)-posy)/r;
                *accz = *accz + force * ((pos[j][2] + cor_z * 34.78 )-posz)/r;
            }
          }

}

}

return potential;
}
```

Algo in above Image reduce Running time significantly by Skipping irrelevent scenarios.

Velocity verlet Algorithm :

```
float update_all(float * posx, float * posy, float * posz, float * velx, float  * vely
{
  float kinetic;
  float mbk = 4.8443;

  *posx = *posx + *velx * delta_time + accx * delta_time * delta_time/2;
  *posy = *posy + *vely * delta_time + accy * delta_time * delta_time/2;
  *posz = *posz + *velz * delta_time + accz * delta_time * delta_time/2;

  // update position with periodic box condition
  if (*posx > 34.78) *posx = *posx - 34.78;
  if (*posx < 0) *posx = *posx + 34.78;
  if (*posy > 34.78) *posy = *posy - 34.78;
  if (*posy < 0) *posy = *posy + 34.78;
  if (*posz > 34.78) *posz = *posz - 34.78;
  if (*posz < 0) *posz = *posz + 34.78;

  *velx = *velx + (accx + last_accx) * delta_time/2;
  *vely = *vely + (accy + last_accy) * delta_time/2;
  *velz = *velz + (accz + last_accz) * delta_time/2;

  kinetic = mbk * (pow(*velx, 2) + pow(*vely, 2) + pow(*velz, 2))/2000;

  return kinetic;

}
```

3 Formula used

- **Leonard-Jones potential :**

$$V_{\mathrm{LJ}} = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$$

- **Radial pair Distribution :**

$$g(r) = (V/N)[n(r)/4\pi r^2 \Delta r].$$

- **Temperature calculation :**

$$T = \frac{M}{3Nk_B} \sum_{i=1}^{N} \mathbf{v}_i^2 ,$$

- **Velocity auto-correlation :**

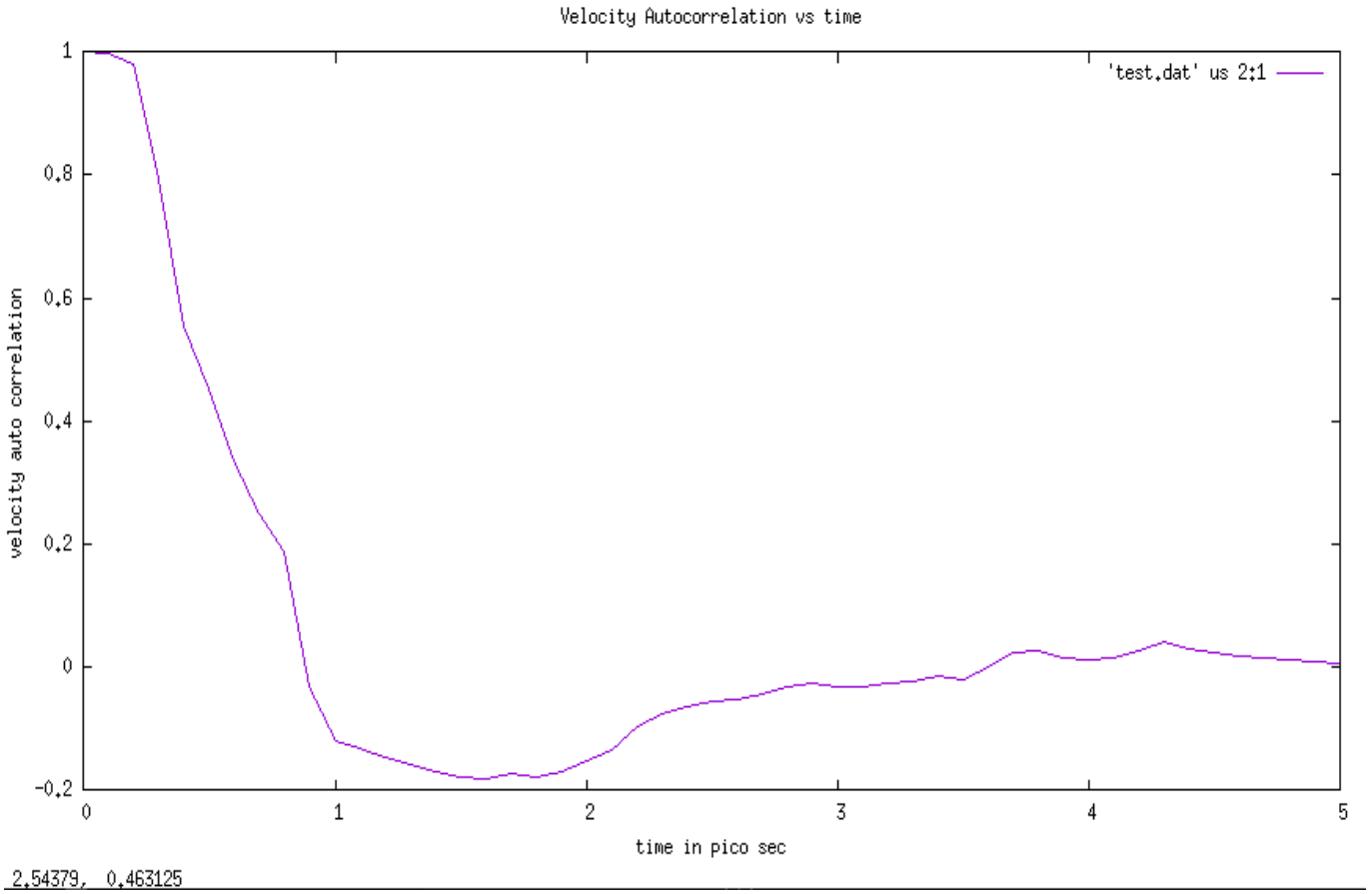
$$\langle \mathbf{v}(0) \cdot \mathbf{v}(t) \rangle = \frac{1}{N} \sum_{i=1}^{N} \mathbf{v}_i(0) \cdot \mathbf{v}_i(t) .$$

4 Results

4.1 Velocity Autocorrelation :

Velocity autocorrelation is a tool to measure diffusion coefficient in MD simulations.
In velocity correlation we take velocity at a time t0 and we measure correlation of velocity withy time.
We get negative value of velocity correlation after sometime which generally comes due to back scatering of atoms, Also after long time all correlation fades away and it tends to zero.

In this experiment I have taken t0 as t = 0, I have moniterd it till t = 5 pico seconds. We can see that my plots qualitatively matches with Rahmans paper but in Rahmans paper it comes to zero at nearly .5 Pico seconds but in my simulation it comes to zero around .75 seconds.

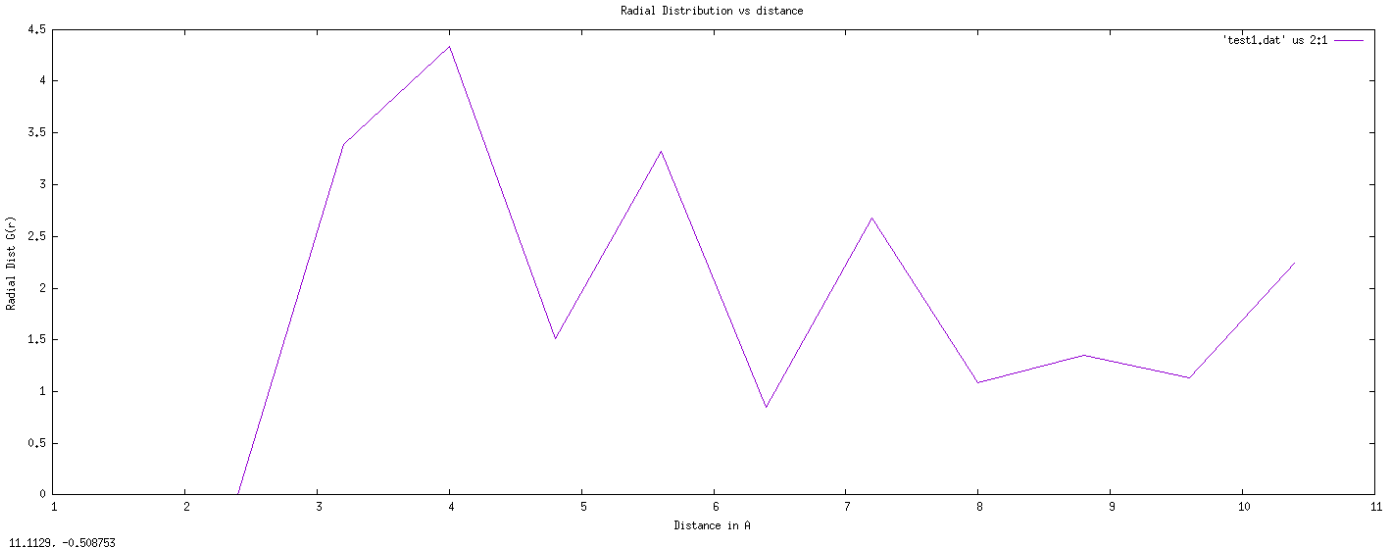


Above picture shows Velocity auto-correlation with time.

4.2 Radial Pair Distribution :

Radial Pair Distribution is a tool of molecular dynamics which describes how density varies as a function of distance from a reference particle. It can also be used to study phage-change behaviour of a material.

In this experiment I have taken delta-r as 0.8 A.
To calculate radial pair distribution I have taken an atom randomly and find the radial distribution surrounding that atom. We can see that In Rahmans paper, we can see the first peak at 3.7 A and in my simulation results it coming to be around 4A "*there is a peak near 3.7 too.*" Also we can see that second peak in rahman's paper is around 7A but in my simulations it's around 6A. it matches with rahman paper's results qualitatively with some error in quantative data.



Above picture shows radial distribution function.

References

[1] A. Rahman, Physical Rev.136, A405 (196) Motion of Atoms in Liquid Argon
http://theo.ism.u-bordeaux.fr/J-C.Soetens/doc/4TCH914U/Rahman_PhysRev_1964.pdf

[2] Molecular Modeling by A. R.Leach

[3] Understanding Molecular Simulation by Frenkel and Smith