MSE760 Assignment 2

Lijing Yang

Mechanical Engineering Department, University of Wisconsin – Madison

March 15, 2018

1 Force Calculation

With a slightly larger preset lattice constant 5.7Å, a constant volume simulation is set up for Ar crystal. On the basis of previous LJ energy calculation, forces for x, y and z coordinates are calculated in the same subroutine of code:

```
// calculate LJ energy
void LJEnergyForce(int xdim, int ydim, int zdim, double a_red, int PBC, int step,
                   double* arrayX, double* arrayY, double* arrayZ,
                   double* arrayXij, double* arrayYij, double* arrayZij,
                   double* forceXij, double* forceYij, double* forceZij, double* LJenergy){

  int numOfAtoms = 4 * xdim * ydim * zdim;
  double distSquare = 0.0;
  double LJ = 0.0;

  if (PBC)
  {
  addPBC(xdim, ydim, zdim, xdim, a_red, arrayX, arrayXij);
  addPBC(xdim, ydim, zdim, ydim, a_red, arrayY, arrayYij);
  addPBC(xdim, ydim, zdim, zdim, a_red, arrayZ, arrayZij);
  }
  else
  {
  noPBC(xdim, ydim, zdim, xdim, a_red, arrayX, arrayXij);
  noPBC(xdim, ydim, zdim, ydim, a_red, arrayY, arrayYij);
  noPBC(xdim, ydim, zdim, zdim, a_red, arrayZ, arrayZij);
  }
  for (int i = 0; i < numOfAtoms; i++) {
      double forceX = 0.0;
      double forceY = 0.0;
      double forceZ = 0.0;
      for (int j = 0; j < numOfAtoms; j++) {
          double Xij = arrayXij[j + i * numOfAtoms];
          double Yij = arrayYij[j + i * numOfAtoms];
          double Zij = arrayZij[j + i * numOfAtoms];
          distSquare = distanceSquare(Xij, Yij, Zij);

          if (distSquare != 0.0){
            forceX += 48.0 * Xij * (pow((1.0 / distSquare), 7) - 1 / 2 * pow((1.0 /
                distSquare), 4));
            forceY += 48.0 * Yij * (pow((1.0 / distSquare), 7) - 1 / 2 * pow((1.0 /
                distSquare), 4));
            forceZ += 48.0 * Zij * (pow((1.0 / distSquare), 7) - 1 / 2 * pow((1.0 /
                distSquare), 4));
```

```
            }
        forceXij[i] = forceX;
        forceYij[i] = forceY;
        forceZij[i] = forceZ;
    } // end of for loop

    for (int i = 0; i < numOfAtoms - 1; i++) {
      for (int j = i + 1; j < numOfAtoms; j++) {
        double Xij = arrayXij[j + i * numOfAtoms];
        double Yij = arrayYij[j + i * numOfAtoms];
        double Zij = arrayZij[j + i * numOfAtoms];
        distSquare = distanceSquare(Xij, Yij, Zij);

        LJ += 4.0 * (pow((1.0 / distSquare), 6) - pow((1.0 / distSquare), 3));
      }
    } // end of for loop
    LJenergy[step] = LJ;
} // end of function
```

The calculated forces of first 50 atoms associated with $10 * 10 * 10$ number of unit cells are listed as following, the units are all in Newton.

```
0 -3.774443e-02 -3.774424e-02 -3.774423e-02
1 -3.774428e-02 -3.774412e-02 3.774577e-02
2 -3.774447e-02 -3.774434e-02 3.774487e-02
3 -3.774450e-02 -3.774431e-02 -3.774786e-02
4 -3.774446e-02 -3.774430e-02 -3.774486e-02
5 -3.774450e-02 -3.774431e-02 3.774488e-02
6 -3.774443e-02 -3.774427e-02 -3.775065e-02
7 -3.774452e-02 -3.774433e-02 -3.774486e-02
8 -3.774448e-02 -3.774432e-02 3.774485e-02
9 -3.774445e-02 -3.774432e-02 3.774485e-02
10 -3.774440e-02 3.774583e-02 -3.774412e-02
11 -3.774428e-02 3.774576e-02 3.774577e-02
12 -3.774447e-02 3.774578e-02 3.774487e-02
13 -3.774453e-02 3.774574e-02 -3.774785e-02
14 -3.774443e-02 3.774571e-02 -3.774486e-02
15 -3.774447e-02 3.774573e-02 3.774488e-02
16 -3.774438e-02 3.774582e-02 -3.775077e-02
17 -3.774449e-02 3.774574e-02 -3.774486e-02
18 -3.774448e-02 3.774576e-02 3.774485e-02
19 -3.774448e-02 3.774573e-02 3.774485e-02
20 -3.774435e-02 3.774488e-02 -3.774432e-02
21 -3.774425e-02 3.774484e-02 3.774575e-02
```

22 −3.774442e−02 3.774487e−02 3.774487e−02
23 −3.774447e−02 3.774482e−02 −3.774789e−02
24 −3.774438e−02 3.774486e−02 −3.774486e−02
25 −3.774445e−02 3.774482e−02 3.774488e−02
26 −3.774435e−02 3.774492e−02 −3.775054e−02
27 −3.774446e−02 3.774486e−02 −3.774486e−02
28 −3.774442e−02 3.774488e−02 3.774485e−02
29 −3.774443e−02 3.774483e−02 3.774485e−02
30 −3.774445e−02 −3.774783e−02 −3.774436e−02
31 −3.774432e−02 −3.774788e−02 3.774577e−02
32 −3.774452e−02 −3.774787e−02 3.774487e−02
33 −3.774457e−02 −3.774779e−02 −3.774775e−02
34 −3.774448e−02 −3.774791e−02 −3.774486e−02
35 −3.774452e−02 −3.774790e−02 3.774488e−02
36 −3.774443e−02 −3.774783e−02 −3.775052e−02
37 −3.774459e−02 −3.774788e−02 −3.774486e−02
38 −3.774453e−02 −3.774789e−02 3.774485e−02
39 −3.774453e−02 −3.774789e−02 3.774485e−02
40 −3.774437e−02 −3.774483e−02 −3.774432e−02
41 −3.774422e−02 −3.774488e−02 3.774575e−02
42 −3.774444e−02 −3.774484e−02 3.774487e−02
43 −3.774445e−02 −3.774490e−02 −3.774789e−02
44 −3.774441e−02 −3.774488e−02 −3.774486e−02
45 −3.774445e−02 −3.774489e−02 3.774488e−02
46 −3.774438e−02 −3.774480e−02 −3.775054e−02
47 −3.774446e−02 −3.774485e−02 −3.774486e−02
48 −3.774445e−02 −3.774484e−02 3.774485e−02
49 −3.774446e−02 −3.774489e−02 3.774485e−02
50 −3.774431e−02 3.774489e−02 −3.774432e−02

It was found that with the increase of number of unit cells in each dimension, the magnitude of force is decreasing to approach zero.

2 Calculated temperature

Temperature is calculated based on the given temperature (i.e. 300K) and initialized random velocities. With a comparison between the temperature calculated from velocities vs. given temperature, the uniform distribution of random velocities can be verified.

```
// calculate average temperature
void calcAveTemp(int xdim, int ydim, int zdim,
                 double givenTemp_red,
                 double* arrayVx, double* arrayVy, double* arrayVz){
  int numOfAtoms = 4 * xdim * ydim * zdim;
```

```
6   double v0 = sqrt(3 * givenTemp_red);
7   double sumTemp = 0.0;
8
9   for (int i = 0; i < numOfAtoms; i++) {
10    double Vx = v0 * arrayVx[i];
11    double Vy = v0 * arrayVy[i];
12    double Vz = v0 * arrayVz[i];
13    double VSquare = Vx * Vx + Vy * Vy + Vz * Vz;
14    sumTemp += VSquare;
15    arrayVx[i] = Vx;
16    arrayVy[i] = Vy;
17    arrayVz[i] = Vz;
18    }
19    sumTemp /= 3 * numOfAtoms;
20    if (fabs(sumTemp - givenTemp_red) > 1e-3){
21      printf("Error! Caculated average temperature is not the same as given
          temperature\n");
22      printf("Caculated average temperature is %f, given temperature is %f\n",
          sumTemp, givenTemp_red);
23    }
24    printf("Caculated average temperature in reduced unit is %f\n", sumTemp);
25    printf("Caculated average temperature in SI unit is %f (K)\n", sumTemp *
        EPSILON / KB);
26  }
```

3 Energy plots

Fig. 1, Fig. 2 and Fig. 3 demonstrate the variation of kinetic energy, potential energy and total energy with different time step sizes. It's clear to see that the total energy reaches a constant value after about 30 time steps, when the system turns into equilibrium. With the initial temperature of 300K, kinetic energy was large at the beginning and converges to about half of the initial value when the system is equilibrated.

It's interesting to see that change of time step size doesn't influence the results of energy as what happened to the MD simulation with LAMMPS in Lab 1. Convergence can still be maintained with step size dT=0.02 in reduced units. Recall that in Lab 1 when using dT=50fs (dT=0.0233 in reduced units), the total energy would not be able to converge. This inconsistency might imply some problems with the proposed code.

In addition, compared with the plots from Lab 1, the proposed code shows a drop of total energy at the beginning on contrast with the constant total energy from LAMMPS. It was first suspected that the reason might be different initial temperature (300K vs. 250K), however,

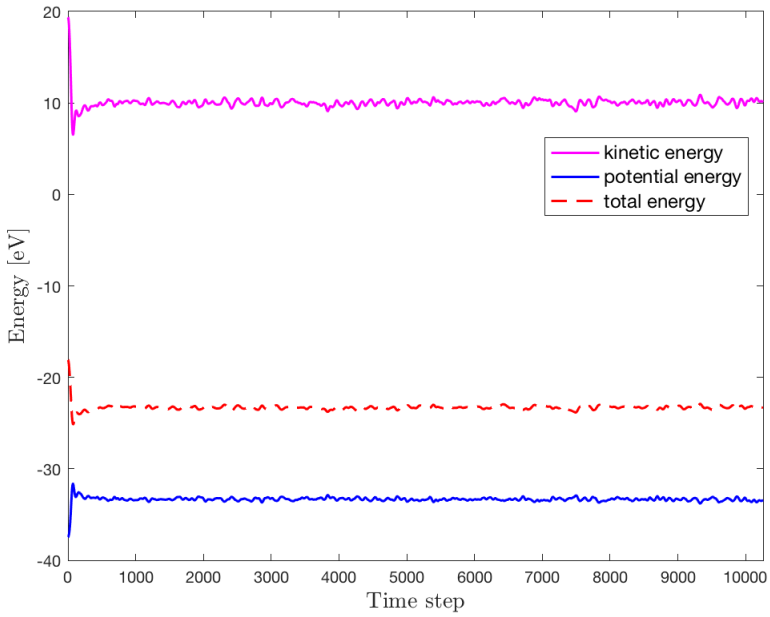

Figure 1: Energy of $5 * 5 * 5$ dimension with time step dT=0.001 in reduced unit

a test at initial temperature 250K preserves the drop of total energy. Another observation is that, at the end of the simulation, when calculating the temperature with final velocities, it is at about 150K instead of 250K as an expected equilibrium temperature.

Therefore, the proposed code might still have issues that interfere the correctness of simulation, which need to be debugged later on.

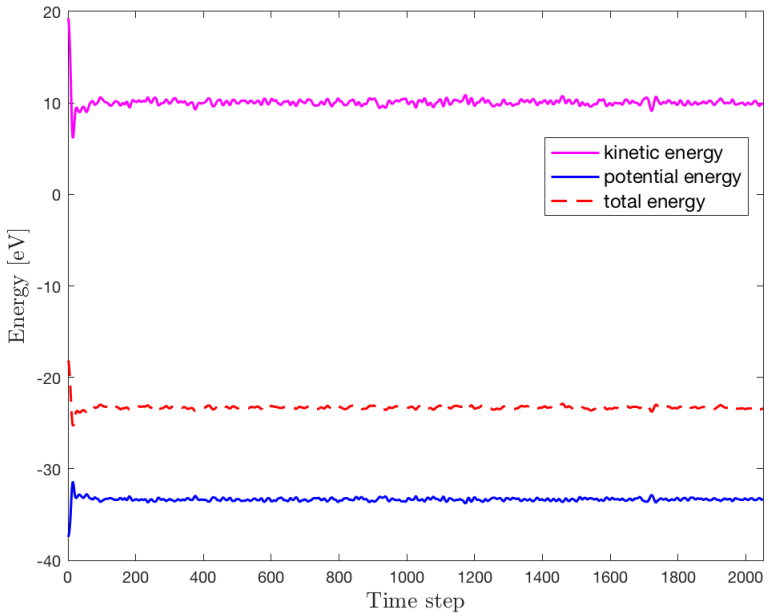

Figure 2: Energy of $5 * 5 * 5$ dimension with time step dT=0.005 in reduced unit

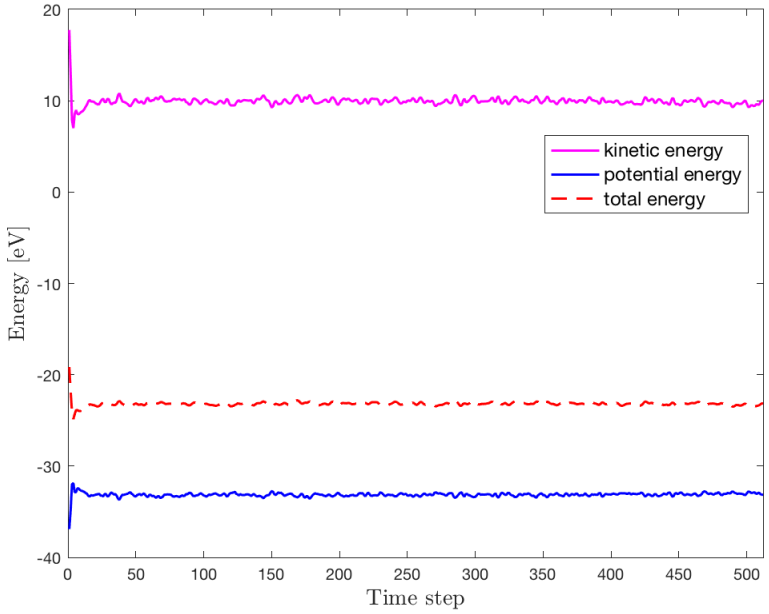

Figure 3: Energy of $5 * 5 * 5$ dimension with time step dT=0.02 in reduced unit

4 Radial Distribution Function g(r)

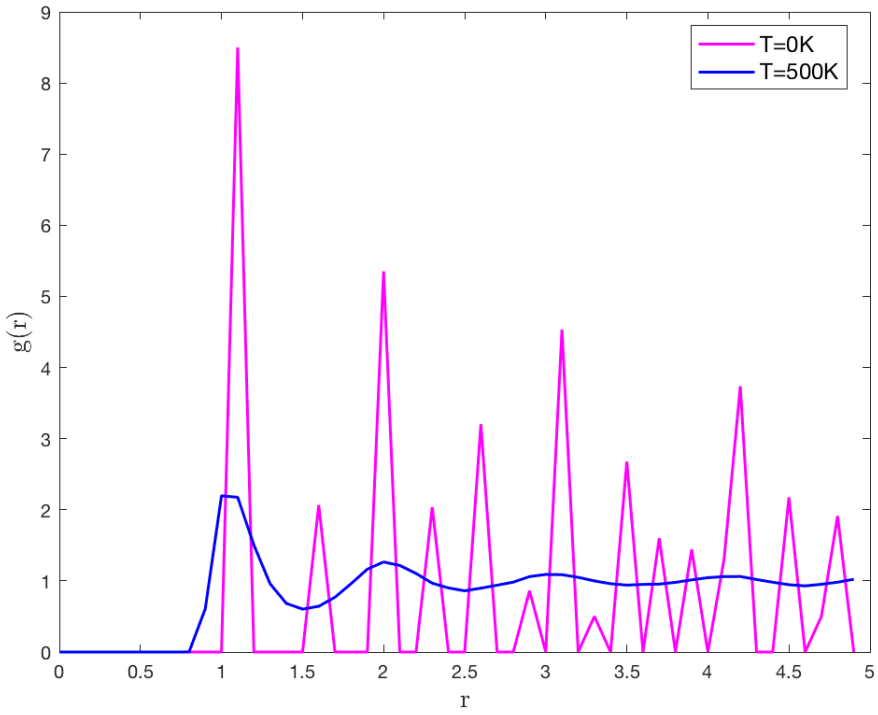

Figure 4: Radial Distribution Function g(r) at T=0K and T=500K after running 50 time steps

Fig. 4 indicates that at 0K Ar crystal is at solid state with peaks representing concentration at particularly favored separation distance (regular spacing). Meanwhile, at 500K temperature, Ar is at liquid state with irregular spacing and smooth variation, also normalized to be close to 1.

5 Usage of code

To compile, use:

```
gcc MDRun.c -std=c99 -lm
```

To run, use:

```
./a.out 5 5 5 0.001
```

where the first 3 input arguments specify the number of unit cells in each dimension, and the 4th input argument specify the time step, whereas the total time is hard-coded to be $2.2e-11$(s). Flags can be set up for the printout of position, energy, force and g(r) results.