Simulation Methods in Physics I

Worksheet 3: Molecular Dynamics 2 and Observables

Students:	Michael Marquardt	Cameron Stewart
matriculation numbers:	3122118	3216338

1 Command line parameters

For the simulation different command line parameters were defined. In the following code blocks you can see them. They will be explained during the report.

Code block 1: Commands for ljsim.py script: src/ljsim.py

```
# command line arguments
parser = argparse.ArgumentParser()
parser.add_argument("--cont", type=double, help="continue calculation with for cont further time")
parser.add_argument("--time", type=double, help="How long do you want to run the simulation? | default time=10s")
parser.add_argument("--tstat", type=double, help="Uses a thermostat with a given temperature")
parser.add_argument("--warm", type=double, help="Use the warming up | pass force")
parser.add_argument("--ctstat", type=double, help="continue for the simulation with tstat")
parser.add_argument("--cwarm", type=double, help="continue for the simulation with warm")

args = parser.parse_args()
```

Code block 2: Commands for ljanalyze.py script: src/ljanalyze.py

```
# command line arguments
parser = argparse.ArgumentParser()
parser.add_argument("--M", type=int, default=10, help="window size | default: M=10")
parser.add_argument("--datafile", type=str, default='../dat/ljsim.dat', help="datafilename | default: '../dat/ljsim.dat'")
parser.add_argument("--teq", type=double, help="equilibration time [s] | calculates averages after equilibration time")
parser.add_argument("--tlim", type=double, nargs='+', help="type in the time limits you want to plot")

args = parser.parse_args()
```

2 Restart the program where left it

First we want to change the programm in a way, that it is able to restart the simulation where it ended last time. The first thing that we have to do is to store the necessary information about the end state of the system, the position x and velocity v of each particle, into the file ljsim.dat. This happens in code block 3. Furthermore there are the new variables Ts and Ps. The represent the temperature and pressure of the system over time and will be explained in the next chapter.

Code block 3: Storing data script: src/ljsim.py

```python
# write out simulation data
print("Writing simulation data to {}.".format(datafilename
   ))
datafile = open(datafilename, 'w')
pickle.dump([ts, Es, Ts, Ps, x, v], datafile)
datafile.close()
```

Now this data must be read by ljsim.py. Therefore we define a command line parameter −−cont which accepts as argument the time for which the simulation should be continued (code block 1). If the parameter is not used the simulation will start new. Furtehrmore there is another argument −−time which takes the simulationtime for a new simulation. The code in code block 4 does exactly this.

Code block 4: Continue simulation script: src/ljsim.py

```python
# Import previous data
if args.cont:
        # open datafile
        datafile = open(datafilename,'r')
        ts, Es, Ts, Ps, x, v = pickle.load(datafile)
        datafile.close()

        # length of run
        t = ts[-1]
        tmax = t+args.cont
        step = 0

else:
        # length of run
        if args.time:
                tmax = args.time
        else:
                tmax = 10.0
```

3 Calculating Temperature and Pressure

3.1 Expanding Energy Calculation

First the energy calculation should be changed in order to also get the kinetic and potential energy. This is done by changing the compute_energy() function call and the parameter Es in ljsim.py (code block 5).

Code block 5: $E_{\text{kin}} + E_{\text{pot}}$ script: src/ljsim.py

```python
    if step % measurement_stride == 0:
        E_pot, E_kin, E_tot = compute_energy(x, v)
        T = 2*E_kin/(3*N)
        P = compute_pressure(E_kin, x)
        print("t={}:\n\tE_pot={}\n\tE_kin={}\n\tE_tot={}\n
           \tT={}\n\tP={}".format(t, E_pot, E_kin, E_tot,
           T, P))

        ts.append(t)
        Es.append([E_pot,E_kin,E_tot])
        Ts.append(T)
        Ps.append(P)
```

3.2 Calculating the Temperature

The temperature of the system can be calculated from the kinetic Energy E_kin like in code block 5 shown. This is a single scalar calculation and therefore can be done in python.

3.3 Calcating Pressure

Derivation of the pressure

The virial of a system is defined as:

$$G = \sum_{i}^{N} \boldsymbol{p}_i \cdot \boldsymbol{r}_i \tag{1}$$

In the case

$$0 = \left\langle \frac{\mathrm{d}G}{\mathrm{d}t} \right\rangle = \left\langle \sum_{i}^{N} \frac{\boldsymbol{p}_i^2}{m_i} \right\rangle + \sum_{i}^{N} \langle \boldsymbol{F}_i \cdot \boldsymbol{r}_i \rangle \tag{2}$$

the following equation can be derived:

$$-\sum_{i}^{N} \langle \boldsymbol{F}_i \cdot \boldsymbol{r}_i \rangle = 2 \left\langle \sum_{i}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} \right\rangle = 2 \langle E_{\text{kin}} \rangle = 3Nk_BT \overset{\text{id. gas}}{=} 3PV \tag{3}$$

For an pair interaction (like in the simulation) let the forces $\boldsymbol{f}_{ij}$ and the vectors $\boldsymbol{r}_{ij}$ be defined as follows:

$$\boldsymbol{r}_{ij} = \boldsymbol{r}_j - \boldsymbol{r}_i \tag{4}$$

$$\boldsymbol{f}_{ij} = f_{\mathrm{lj}}(\boldsymbol{r}_{ij}^2) \cdot \frac{\boldsymbol{r}_{ij}}{|\boldsymbol{r}_{ij}|} \tag{5}$$

With this knowldege we can derive the pressure of the system:

$$\begin{aligned}
P &= P_{\text{id. gas}} + P_{\text{interaction}} && (6)\\
&= \frac{Nk_BT}{V} + \frac{1}{3V}\sum_i^N \langle \boldsymbol{F}_{i,\text{interaction}} \cdot \boldsymbol{r}_i \rangle && (7)\\
&= \frac{1}{3V}\left[\left\langle \sum_i^N \frac{\boldsymbol{p}_i^2}{2m_i} \right\rangle + \left\langle \sum_{i,j\neq i}^N -\boldsymbol{f}_{ij} \cdot \boldsymbol{r}_i \right\rangle\right] && (8)\\
&= \frac{1}{3V}\left[\left\langle \sum_i^N \frac{\boldsymbol{p}_i^2}{2m_i} \right\rangle + \left\langle \sum_{i,j>i}^N \boldsymbol{f}_{ij} \cdot \boldsymbol{r}_j - \boldsymbol{f}_{ij} \cdot \boldsymbol{r}_i \right\rangle\right] && (9)\\
&= \frac{1}{3V}\left[\left\langle \sum_i^N \frac{\boldsymbol{p}_i^2}{2m_i} \right\rangle + \left\langle \sum_{i,j>i}^N \boldsymbol{f}_{ij} \cdot \boldsymbol{r}_{ij} \right\rangle\right] && (10)\\
&= \frac{1}{3V}\left[2\langle E_{\text{kin}}\rangle + \left\langle \sum_{i,j>i}^N \boldsymbol{f}_{ij} \cdot \boldsymbol{r}_{ij} \right\rangle\right] && (11)
\end{aligned}$$

Pressure in Cython

Becaus of the vectorial calculations for each particle the calculation has to be done in C if it should be fast. Therefor the function c_compute_pressure() is written in c_lj.cpp.

Code block 6: E_{kin} + E_{pot} script: src/c_lj.cpp

```cpp
double c_compute_pressure(double E_kin, double* x){
  double rij[3];
  double fij[3];
  double interaction = 0.0;

  // for (int i = 1; i < N; i++)
  //   for (int j = 0; j < i; j++) {

  // add up fij*rij
  vector<int>::iterator it = verlet_list.begin();
  vector<int>::iterator end = verlet_list.end();
  while (it != end) {
    int i = *it;
    ++it;
    int j = *it;
    ++it;

    if (i!=j){
      minimum_image(x, i, j, rij);
      compute_lj_force(rij,fij);

      for (int k = 0; k < 3; k++)
        interaction += fij[k]*rij[k];
    }
  }
  return (2.*E_kin+interaction)/(3*L*L);
}
```

4 Molecular Dynamics at a Desired Temperature

For the *velocity rescaling* thermostat we can derive the rescaling-factor f_{re} from equation (12).

$$\begin{aligned}
\frac{3}{2}k_B T_0 &= \frac{E_{\mathrm{kin},0}}{N} && (12)\\
&= \frac{1}{N}\sum_{i=1}^{N}\frac{(f_{\mathrm{re}}\cdot \boldsymbol{v}^{(i)})^2}{2m} && (13)\\
&= f_{\mathrm{re}}^2\frac{E_{\mathrm{kin}}}{N} && (14)\\
&= f_{\mathrm{re}}^2\frac{3}{2}k_B T && (15)\\
f_{\mathrm{re}} &= \sqrt{\frac{T_0}{T}} && (16)
\end{aligned}$$