Notes on thermodynamics and the Lennard-Jones potential

1 The dynamics

This simulation is somewhat similar in character to your vibrating string: you have a set of N objects that exert forces on each other. The only three differences are:

- The force isnt given by Hookes law; its given by the Lennard-Jones potential
- Instead of each mass feeling a force from two neighbors, it feels a force from all of the masses
- There are also reflective walls

1.1 Finding the force from the Lennard-Jones potential

Note: This derivation differs slightly from what we did in class on Tuesday. The only differences are overall constants that can be absorbed into k and r_0, so in the end, the difference doesn't matter. However, this is the "standard" formulation here.

The potential is

$$V(r) = 4k\left[\left(\frac{r_0}{r}\right)^{12} - \left(\frac{r_0}{r}\right)^{6}\right]$$

You know that the force, in general, is given by the negative derivative of the force ("things roll downhill" – if the potential is decreasing in the +x direction, then things will feel a force that way). In symbols,

$$F(r) = -\frac{\partial V}{\partial r}$$

Thus we differentiate the Lennard-Jones potential to get the force:

$$F(r) = k\left[48\frac{r_0^{12}}{r^{13}} - 24\frac{r_0^{6}}{r^{7}}\right]$$

We can absorb a factor of 24 into k to make this cleaner:

$$F(r) = k\left[2\frac{r_0^{12}}{r^{13}} - \frac{r_0^{6}}{r^{7}}\right]$$

You can then apply the familiar "unit vector trick" to get the components of the force.

1.2 Dealing with the force from many other particles

This is simple – you just add them up, since the net force is the sum of all the forces. However, since each particle feels $N-1$ forces, you have to do this with a `for` loop. This means that whenever you need to evaluate forces you have two nested loops: for each particle (one loop), calculate the force that each other particle (another loop) exerts on it.

1.3 Reflective walls

The easiest way to do this is to flip the sign of the velocity in the offending direction if the particle is both

- "Inside" the wall
- Moving further into the wall

Thus, you might write something like the following if the left-hand wall of your box is at x=0:

```
if (vx<0 && x<0) vx = -vx;
```

2 The initial conditions

You cant control the temperature directly, since your initial conditions will of necessity be far from equilibrium (since you dont know what equilibrium looks like a priori). In order to have access to the widest range of temperatures, I find that the best thing to do is to set the particles up in an arrangement that gives them as little potential energy as possible. Then you can add as much kinetic energy as you want by playing with the initial velocities.

The easiest thing to do to achieve a fairly low initial potential energy is to set the particles up in a square grid separated by a distance corresponding to the minimum of the Lennard-Jones potential. That distance is:

$$r_{\mathrm{eq}} = 2^{1/6} r_0$$

which is just the distance at which the force is zero. This will give you the initial conditions in space. To control the initial kinetic energy, just give them random values for v_x and v_y up to some variable maximum. To generate random numbers, use the function `drand48()`, which returns a random number between 0 and 1; you can multiply the result by a constant to get random numbers over some other range. To use this function, you must write

```
#include <stdlib.h>
```

at the top of your code.

3 Thermodynamic measurements

Remember that you are measuring the thermodynamic observables (temperature and pressure) in the "thermodynamic average": averaged over all particles and over a significant time interval.

I find that the best way to do this is to have accumulator variables ("total impulse" and "total kinetic energy"); every timestep or wall-collision, add the appropriate value to them. Then, at the end of the interval, when were ready to measure the thermodynamic quantities, divide the accumulator variables by the relevant denominator to get the thermodynamic average.

3.1 Measuring the pressure

Pressure is conventionally force per unit area. However, since we are in two dimensions with one-dimensional walls, the pressure on the walls has dimensions force per unit length. Our task is to figure out how to compute this from the dynamics.

Whenever a particle hits a wall, its velocity in one dimension is reversed. This means that there is some impulse delivered from the wall to the particle.

Suppose you have the following code for reflections off of the left-side wall:

```
if (vx[i] < 0 && x[i] < 0)
{
  vx[i] = -vx[i];
}
```

Whenever this happens, note that the change in momentum of the particle (the impulse that the wall delivers) is `2*m*vx[i]`. Thus, to keep track of the total impulse, you should keep a total of the momentum change in the particles that hit the walls.

Note that for our purposes here impulse is a scalar; you will want to add contributions from both "particle hitting left wall is pushed to the right and particle hitting right wall is pushed to the left".

After we wait long enough, we need to convert from total impulse during that interval to average pressure during that interval. This is pretty simple:

- Force F is impulse J over time: $F = J/t$
- Pressure P is force F over length: $P = F/L = J/(tL)$

Notice that t in the above equations is the *amount of time that you accumulated impulse measurements.*

4 Measuring the temperature

The equipartition theorem from statistical mechanics says that

$$\left\langle \frac{1}{2} m v_i^2 \right\rangle = \frac{1}{2} kT$$

that is, the average kinetic energy for each dimension is equal to one-half times kT. (k is Boltzmann's constant, which follows T around like a puppy in physics; it converts from Kelvins to a unit of energy.)

The equipartition theorem tells us the conversion between temperature and average kinetic energy.

In two dimensions, we have

$$\left\langle \frac{1}{2} m v_x^2 + \frac{1}{2} m v_y^2 \right\rangle$$

but this left side is just the kinetic energy, so we have the simple relation

$$kT = \langle KE \rangle$$

In my code I have an accumulator variable for the accumulated kinetic energy. Every timestep, I add the kinetic energy to this accumulator variable; when I am ready to take a temperature measurement, I just divide this "total energy" variable by the product of the number of timesteps and the number of particles to get average kinetic energy per particle; this is thus also the temperature.

Note that, in three dimensions, we have

$$kT = \frac{2}{3} \langle KE \rangle$$

which is the formula you learned in chemistry class.

4.1 Equilibriation, graphing, and analysis

As we have discussed, the thermodynamic data you measure in the beginning of your simu- lation is very far from equilibrium and thus should be excluded.

A reasonable thing to do is to plot some quantity (the temperature, the pressure, or their ratio) vs. time and look for a plateau; you should exclude data taken before that point. Note that the time required to reach equilibrium depends on the temperature! There is no hard and fast rule for doing this; it is something that computational physicists often bicker about. See the classroom demo. You can then average everything after that point together.

The volume and kind of data youll be dealing with is complex enough that the data analysis can get unwieldy if youre not careful. You might want to use a spreadsheet (such as Excel) for your analysis. You can also do it in C by using input redirection, if you want.

5 Dealing with long jobs and expensive computations

Some of your code may take quite a while to run. Thats okay. (Notably, yours will be slower than what I show on the board; I have some magic to neglect interactions between distant particles that I dont ask you to reproduce. If you're curious, ask me about it! Implementing this will count as part of your project.)

Come talk to me if you're wanting to do a very long calculation and I can help you with ways to make it more convenient.