CSCI 2951G: Assignment 1

Paul Valiant

September 13, 2012

Due: Wednesday, September 19.

Turn in: Matlab code needed for each problem; any outputs (numerical or plots) needed to justify the answer; a (brief) explanation of what you did for each problem. Also, answer the following questions, at whatever length you feel is appropriate: 1) What is the most confusing thing from the course so far? 2) What is the most intriguing thing about the course so far? 3) Do you have any additional thoughts about protein folding that you want to contribute to the discussion?

Matlab guidance: Some of these problems require slightly more complicated data processing than we have seen so far in class. Remember that you can always save several matlab commands to a .m file and run/debug them interactively. Type `edit` to open the Matlab editor.

Matlab has some of the best built-in help that I have seen. Type `help` to get a list of topics, and towards the top will be some of the most useful things to get further help on: `help ops`, `help lang`, `help elmat`, `help elfun`.

Also, google for help, or ask other people in the class, though each of you must write up the assignment separately, and must say who you collaborated with.

1 Problems

For each of these problems, construct a protein of your choice, which has at least 10 amino acids. You can choose whether to use implicit water or not (the simulation runs up to 7 times faster without implicit water). See the other handout from today for all the features of the `proteins` software; Section 4 outlines the graphical interface.

1.1 Problem 1:

Start the simulation, as `out=proteins(d2.p,d2.edges,d2.types,p2);` and after unpausing it with the spacebar, lower the friction to 0 (the bottom slider), leave the temperature at 300 kelvins, run it for a little bit, and then right click to output 1000 velocities. Pick an atom and compute its average kinetic energy from the simulation. (Find its mass, from `p2.weights`, and remember to include both the x, y, and z velocities. You can compute the mean in Matlab with the `mean` command, or just use `sum` instead and divide by 1000.) Divide this kinetic energy by the `simstep` parameter, which is .0004 by default unless you have changed it. Recall from class that the average kinetic energy at temperature T is $\frac{1}{2}Tk_B$ per degree of freedom, where k_B is Boltzmann's constant. Thus since an atom can move in 3 dimensions, the energy you just computed should equal roughly $\frac{3}{2}Tk_B$, where $T = 300$; from this, estimate what Boltzmann's constant is, in our units. (To see what Boltzmann's constant actually is, go to google and enter `boltzmann constant * avogadro's number /1 kcal`. If your estimate was slightly too high, then maybe the simulation needs to be run longer for it to equilibrate.

1.2 Problem 2:

Do the same thing as for Problem 1, except find two atoms with the *same* mass, m, and compute the energy of a fictitious particle whose velocity is the *difference* of their velocities, and whose mass is $\frac{m}{\sqrt{2}}$. This should be the same as the energy calculated above. Draw a histogram of the x coordinate of this velocity – by default, Matlab's `hist` function draws histograms with 10 bins; you can change this number by adding a second argument to the `hist` function: `hist(data,100)` draws a histogram of data with 100 bins. (If your data is along the third dimension, remember to `squeeze` your data before sending it to histogram or plot commands.)

1.3 Problem 3:

Do the same thing as for Problem 1, except for the center of mass of the first 20 atoms (or any 20 atoms of your choice). How can you see which the first 20 atoms are? Run `p2.big=1:20` and then these 20 atoms will be rendered big. More explicitly: if you treat the center of mass of the first 20 atoms as a single particle, of mass the total mass of the 20 particles, `sum(p2.weights(1:20))`, see how its energy behaves, and whether you can estimate Boltzmann's constant this way too. (Computing the weighted average of the velocities of these 20 particles might involve a `for` loop, or you can do trickier Matlab.) Note that it might take 20,000 or more steps before this center-of-mass energy settles down close to its expected value (why is this?). You can store data for the entire 20,000 length run by setting `p2.output=20000`, instead of right clicking to output just 1,000 velocities, though be sure not to include the early velocities in your average, as they may be from before the system has equilibrated.

1.4 Problem 4:

Get output from at least 10,000 steps, where, after a few hundred steps the third slider (friction) is moved to 0, and after about a thousand steps when the temperature (as measured in the top right) looks somewhat close to 300, the top slider (how tightly to control temperature) is also moved to 0. We are now going to see to what degree energy is conserved. The `GGB` field of the output has ten columns, of which the first column records kinetic energy, and the next 7 columns record different components of potential energy. The question is, whether energy is being conserved well or not. You can plot the sum of each row as `plot(sum(out.GGB,2))`. The plot will probably look nicer if you ignore the first several rows of data; for example, to plot only the portion starting with the 1000th row, use `plot(sum(out.GGB(1000:end,:),2))`, where the `end` keyword, when used as the index into a matrix, denotes the last possible index (you can also use `end-1` and other variants, for fancier things). If you zoom in, how much does this measure of energy seem to oscillate?

The reason why this plot is so messy is that kinetic energy is a function of velocity, and potential energy is a function of position, and the leapfrog method effectively evaluates position and velocity at staggered timesteps. In order to get a better sense of energy, we should approximate potential and kinetic energy at the *same* time. To do this, we can take kinetic energy and *interpolate* it to estimate its values at timesteps halfway between the timesteps for which we know its values. Try this. There are two possibilities: kinetic energy needs to be shifted half a timestep *backwards* for it to match up with potential energy, or half a timestep *forwards*. Try both and add them to the potential energy; the right choice will be the one which leads to better energy conservation.

You can also try different values of the `simstep` parameter. For values above .002, there will be very bad energy conservation; for smaller values energy will look increasingly flat.