CSCI 2951G: Assignment 2

Paul Valiant

September 25, 2012

Due: Monday, October 1.

Turn in: A file `myforces.m` that computes the forces on each atom of a molecule, given the positions of the atoms and physics parameters, along with a brief analysis of its performance: pick a sequence of 20 amino acids and describe, roughly, the computational cost of your code, counting (scalar) additions and multiplications as taking time 1, division, square root, trigonometric functions, `exp` and `log` as taking time 100. If your code needs additional explanation to run, please include it, and also explain if you made any unusual choices or discovered anything interesting in the process. As usual, be sure to credit others (from the internet or from class) for ideas you used.

The Problem:

Write a Matlab function `myforces.m` that computes the forces yielded by our energy function:

$$E = \sum_{\text{bonds}} K_r(r - r_{\text{eq}})^2 + \sum_{\text{angles}} K_\theta(\theta - \theta_{\text{eq}})^2 + \sum_{\text{dihedrals}} \frac{V_n}{2}[1 + \cos(n\phi - \gamma)] + \sum_{i<j} \left[\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\epsilon R_{ij}} \right] \quad (1)$$

Two inputs will be provided to your function: 1) the positions of all the atoms, as a matrix `pos` with 3 columns and one row per atom, and 2) a structure specifying physics `phys`, as outlined below (similar to the structure `p2` we have been using, but different in many details).

The fields of the structure `phys` are as follows:

- `bonds`: A four column matrix with each row specifying a term from the first term of Equation 1: the first two entries specify a pair of bonded atoms; the third entry specifies the spring constant K_r; the fourth entry specifies the equilibrium length r_{eq}. (This is exactly as for `p2`.)
- `angles`: A five column matrix with each row specifying a term from the second term of Equation 1: the first three entries specify three atoms bonded in sequence which are to have their angle constrained by a spring; the fourth entry specifies the spring constant K_θ in units relative to radians; the fifth entry specifies the equilibrium angle θ_{eq} in degrees. (This differs from the corresponding entry in `p2` as the atoms are specified by a pair instead of a triple there.)
- `dihedrals`: A seven column matrix with each row specifying a term from the third term of Equation 1: the first four entries specify four atoms, not necessarily bonded in order, whose dihedral angle is denoted ϕ in Equation 1; the fifth entry specifies the multiplier $\frac{V_n}{2}$; the sixth entry specifies the offset γ in degrees; the seventh entry specifies the order n (which will always be an integer).
- `vdw`: A three column matrix with one row per atom, specifying the atom's Lennard-Jones radius ρ_i, its Lennard-Jones coefficient c_i, and its charge q_i. As discussed in Lecture 5, to compute the interaction between a pair of atoms i and j, the radii are added and the coefficients c_i, c_j are combined via their geometric mean $\sqrt{c_i c_j}$ to yield a Lennard-Jones term of $\sqrt{c_i c_j}\left[\left(\frac{\rho_i+\rho_j}{R_{ij}}\right)^{12} - 2\left(\frac{\rho_i+\rho_j}{R_{ij}}\right)^{6}\right]$

where R_{ij} is the distance between atoms i and j. Thus in the fourth term of Equation 1, we have $A_{ij} = \sqrt{c_i c_j}(\rho_i + \rho_j)^{12}$ and $B_{ij} = 2\sqrt{c_i c_j}(\rho_i + \rho_j)^6$. The value of ϵ in the denominator of the last term should be 1/332.0636 (as in, 332.0636 should be in the numerator).

Crucially, these "non-bonded" terms must be computed for every pair of atoms *except* those that are closer than three bond lengths apart (graph theoretically), and for atoms that are exactly 3 bond lengths apart the Lennard-Jones force must be divided by 2 and the electrostatic force must be divided by 1.2

There should be no unit conversion necessary for this assignment (except between degrees and radians as appropriate). The nontrivial part will be taking the gradient of Equation 1 to compute the forces. For some of the terms, you might be able to derive the correct answer with intuition, or by remembering things from your last physics course. The dihedral angle term is messy enough that I will derive the force for you (below). The term for (regular) angles you will have to derive yourself, but be careful not to be misled by your intuition: if an angle is not at equilibrium, this will induce a force on all three of its atoms, not just the two endpoints as you might expect. (As a further sanity check, remember that the *total* force on the system should always be 0.)

Dihedral angles

Given four points in three dimensional space, p_1, p_2, p_3, p_4, to find their dihedral angle ϕ, construct the three vectors $v_1 = p_2 - p_1, v_2 = p_3 - p_2, v_3 = p_4 - p_3$, and then, provided these vectors are all nonzero and v_2 is not collinear with either of the others, ϕ is the angle counterclockwise from $-v_1$ to v_3 if the vectors are rotated so that v_2 points straight at you. Wikipedia lists the formula as $\text{atan2}(|v_2| v_1 \cdot (v_2 \times v_3)\,,\, (v_1 \times v_2) \cdot (v_2 \times v_3))$ where the function `atan2(y,x)` computes the angle, counterclockwise from the x-axis to the point (y, x) – look this function up in Matlab or Wikipedia.

We also need to know the gradient of the dihedral angle expression. Since the dihedral angle is a function of v_1, v_2, v_3, the gradient will have 3 parts: the gradient of the dihedral angle with respect to v_1 equals $G_1 = (v_1 \times v_2)\frac{|v_2|}{|v_1 \times v_2|^2}$ – it makes sense that to change v_1 so as to most increase the dihedral angle you should change it in a direction orthogonal to itself, and orthogonal to the axis v_2, and it also makes sense that the gradient is independent of v_3, independent of the length of v_2 (since v_2 appears twice in the numerator and denominator), and depends inversely on the length of v_1. Correspondingly, the gradient with respect to v_3 equals $G_3 = (v_2 \times v_3)\frac{|v_2|}{|v_2 \times v_3|^2}$. The expression for the gradient with respect to v_2 is more complicated: $G_2 = -(v_1 \times v_2)\frac{v_1 \cdot v_2}{|v_2| \cdot |v_1 \times v_2|^2} - (v_2 \times v_3)\frac{v_2 \cdot v_3}{|v_2| \cdot |v_2 \times v_3|^2}$. Since we actually want the gradients with respect to p_1, p_2, p_3, and p_4, we can use the chain rule to deduce that these four gradients are, respectively, $-G_1, G_1 - G_2, G_2 - G_3$, and G_3. Of course, if the potential energy depends on the dihedral angle as $f(\phi)$ then the chain rule yields that the force on, say, atom 2 is $-f'(\phi)\,(G_1 - G_2)$.

Matlab snippets

If, for fun, you want to run a physics simulation from your force calculations, you could use the following function, which takes as arguments the same `pos` and `phys` arguments as above, but also new arguments `vel` which stores velocities in Angstroms per femtosecond, `masses` for the masses of each atom in atomic mass units, and `timestep` for the timestep in femtoseconds, and outputs `posnew` and `velnew` via the leapfrog method:

```
function [posnew,velnew]=myleapfrog(pos,phys,vel,masses,timestep)
forces=myforces(pos,phys);
velnew=vel+forces./repmat(masses(:),1,3)*timestep*.0004184;
posnew=pos+velnew*timestep;
```

While I would suggest constructing your own test cases as you debug your code, if you want to run your code on real protein data I have included a function `homeworkapplyphys.m` which behaves like the

original `applyphys` function except it outputs a structure in the special form given in this homework, so you can (hopefully!) run your `myforces` function on a position matrix and this physics structure. You could then compare your output directly with the forces output by `proteins`. When setting up for a comparison, make sure to call `applyphys` without implicit water, and with its last line commented out (as I mentioned previously, I accidentally included an experimental feature in the proteins code, which should be removed by commenting out the call to the function `dihedralbump` – comments in Matlab start with the % character.)