# CSCI 2951G: Guide to the `proteins` Code

Paul Valiant

## 1 Loading Data

Loading data to run `proteins` involves four steps, as in the following:

- `d=loaddat;` loads data from disk about each amino acid and related molecules. For example, after running this command, typing `d{10}` will output the data loaded for the 10th amino acid in the data files, arginine.
- `d2=loadprotein(d,{'nasn','leu','tyr','ile','gln','trp','asp','gly','gly','cpro'},'b');` uses the data loaded to `d` to construct a sequence of amino acids, as specified by their 3 letter abbreviations. The final argument is `'b'` to construct them in a straight line (as in a *beta sheet*), or `'a'` to construct them in a helix (as in an *alpha helix*). A 3 letter abbreviation can be preceded by "n" or "c" to indicate that it is at the start or end of the amino acid chain respectively, as amino acids in the end positions are slightly structurally different.
- `p=loadphys;` loads physics parameters from disk which describe, for example, the spring constants between different pairs of bonded atoms, and the angular spring constants between triples of bonded atoms, etc.
- `p2=applyphys(d2,p,'i');` applies the physics just loaded into `p` to the amino acid sequence loaded into `d2`. The final parameter `'i'` is optional, and signifies that an implicit water model should be prepared; otherwise the protein is simulated in a vacuum.

## 2 Standard Inputs

The `proteins` function is typically run as `proteins(d2.p,d2.edges,d2.types,p2);` where: the first argument is a matrix with 3 columns where each row describes the initial position of the corresponding atom in the protein; the second and third arguments generally are not modified by the user, but describe the structure of the protein as it is drawn on screen, including the edges, and the types (colors and sizes of the atoms); the fourth argument is a structure (accessed with the dot "." operator) which contains the physics, and many optional parameters which are described in the next section.

The standard fields contained in `p2` are as follows:

- `weights` contains the weights of each atom, in molecular mass units – hydrogens are roughly 1, carbons are roughly 12, nitrogens are roughly 14, and oxygens are roughly 16. These parameters can be changed for interesting effects; for example, `p2.weights(:)=1;` makes all the atoms have unit mass.
- `bonds` contains the parameters of the "springs" between atoms that model each atomic bond, as in the first term of the energy equation (see the notes from the first lecture). Each row of `bonds` contains four numbers: the indices of the two atoms bonded; the spring constant; and the equilibrium length of the bond, in nanometers. The units of the physics constants here are standard amongst basically all molecular dynamics software: energy is measured in kilocalories per mole; distance is measured in Angstroms. Thus the units of the spring constant here are $kcal/(M\,A^2)$.

- `angles` contains the parameters of the angular "springs". The format is similar to that for `bonds`: the first two entries specify the two *bonds* that specify the angle. (If the two bonds do not share an atom, an error is thrown.) The third parameter specifies the spring constant, in units of kilocalories per mole, per radian squared. The fourth parameter specifies the equilibrium angle, in degrees.

- `dihedrals` contains the parameters specifying the dihedral component of the energy function. The first two parameters specify a pair of *angles* that jointly specify the dihedral angle; the pair of angles must share an edge; if the vertexes at the apex of the two angles are the same, the dihedral is called an *improper* dihedral. The third parameter is a normalizing factor that the force constant is divided by; the fourth parameter is the force constant, in the same units as above for angles. The fifth parameter specifies the equilibrium angle. The sixth parameter specifies the degree of the cosine function that contributes to the energy (check out the equation for energy from the first lecture).

- `VDW` contains parameters for all the "nonbonded" interactions, which includes the Lennard-Jones potential and electrostatic interaction. These interactions are computed for every pair of atoms that are *not* within 2 bonds of each other. (Atoms that are 3 bond lengths from each other have the interaction strengths scaled down, but not to 0, by convention.) The first two columns specify the pair of atoms interacting. The next three columns specify the strengths of the $r^{-12}, r^{-6}$, and $r^{-1}$ terms respectively.

- `implicit` contains extra information needed to implicitly simulate the effect of water: the `types` subfield contains the types of the atoms, and the `charges` subfield denotes the electric charges. (The electric charges are also used to compute the electrostatic forces, which are parameterized in the `VDW` field above; here they are used in a different way.)

- `big` is a list of indices of which atoms to optionally render "big". This behavior can be modified by pressing "b" once `proteins` is running, so that all or none of the atoms are drawn big, which can help visibility.

# 3 Optional Parameters

- `simstep` This parameter controls the timestep of the simulation. The parameter is a bit awkwardly specified: each timestep equals roughly 50 femtoseconds times the square root of `simstep`. Thus for a timestep of 2fs, set `simstep` to be `.0016`. (The default value is `.0004`, for a 1fs timestep.) Even slightly higher than this and the simulation will get unstable, as you can tell by watching whether velocities are blowing up and have been capped, according to whether the "V caps" number in the top left corner of the `proteins` window is climbing.

- `params`: a triple of values specifying the sliders in the bottom right of the screen: first parameter specifies how tightly to control temperature – .01 is default but .04 is probably also reasonable; second parameter is what temperature to aim for, in kelvins – 310, body temperature, is reasonable though by default it starts at 300; third parameter is friction, which should start out high (.5 or .1) if the initial configuration is constructed ad hoc, but should be lowered to 0 soon, or if restarting a previous simulation (via the inputting the position and velocity output by a previous run)

- `pause`: whether to start paused or not; 0 means no.

- `output`: if this is nonzero, then physics is simulated for this many steps, and then the position, velocity, forces, and different kinds of energy are output for each step (though since the simulation uses the leapfrog method, be careful about being effectively off by half a timestep). Alternatively, if you want to output these things from the middle of a run, right click in the `proteins` window and click `Output` and then `1000 Velocities` to output the next 1000 steps.

- `glue`: specifies a list of atoms to glue in place. This can always be modified with the 3d mouse and pressing 'g' and 'G'.

- `velocity`: specify initial velocities of the $n$ atoms with a matrix with $n$ rows and 3 columns.
- `drawevery`: draw the screen every how many iterations of physics? (5 is default)
- `pushscale`: this just scales the effect of the 3d mouse. (1 is default.)
- `screensize`: this should be set to the diagonal size of your screen, in inches, for the 3d stereo to work properly. (14 inches is the default.) If you set this, then you should also right-click in the `proteins` window and properly set the distance from you to the screen.

# 4 The `proteins` GUI

## 4.1 Keyboard Commands

- <esc> quits the program.
- "p" toggles whether the sliders in the bottom right are visible.
- "f" toggles whether the text in the bottom left is visible.
- <enter>: if the 3d cursor is not visible, makes the cursor visible; if it is visible, grabs the nearest atom so that it will be pushed by the 3d mouse. Pressing <shift-enter> allows selection of multiple atoms; selection is ended by pressing <enter> alone.
- "g" glues the nearest atom in place if the 3d cursor is shown; unglues all atoms if atoms are already glued; and otherwise reglues the last glued atom if the cursor is invisible. Pressing shift allows gluing of multiple atoms, as above.
- "b" specifies which atoms will be rendered big: cycles between the atoms specified by the `big` parameter, all atoms, and none of the atoms respectively.
- "c", when the implicit water model is used, displays atom charges via the color and sizes of atoms (big means more highly charged; yellow and blue are opposite charges); pressing "c" again will display atoms at their *effective Born radius*, something computed in the implicit water calculation, though this is only displayed after physics simulation has begun.
- "w" displays the forces due to their nonbonded interactions. If implicit water is being simulated, there are two more options: pressing "w" again displays nonbonded forces damped by the *shielding* effect of water; pressing "w" a third time displays the final component of the implicit water model, capturing the amount highly charged particles are attracted to the nearest region of water.
- "s" is for use with the 3d mouse: when an atom is selected, it displays the nonbonded forces all the other atoms exert on the selected atom.
- <space> pauses or unpauses physics.

## 4.2 Right-Click Mouse Commands

- Distance to Screen: set the distance from you to the screen here, for best 3d stereo.
- Screen Diagonal: set the size of your screen's diagonal here, for best 3d stereo.
- Camera: "Reset" sets the display to 3d stereo, at default zoom and orientation; "Bigger" and "Smaller" make the molecule bigger or smaller respectively; "No Parallax" disables 3d stereo, and "Parallax" enables it.

- Glasses Colors describes the type of 3d glasses you are wearing: "Red-Cyan" is the default. If you are having trouble seeing colors of the atoms, it might be that your left eye is dominant, in which case you should put the glasses on backwards so the cyan lens is over your left eye, and select the last option, "Cyan-Red".
- Ambient Lighting, Diffuse Lighting, and Specular Lighting all specify how the material of the atoms looks.
- Output: the "1000 Velocities" option outputs data from the next 1000 steps of physics.

### 4.3 On Screen Displays

The sliders in the bottom right are also settable programmatically with the `params` argument, and specify from top to bottom: how tightly temperature is controlled – when the system has an energy different from the expected energy at the specified temperature, the velocities are rescaled every time a frame is drawn to be 1% closer to what the energy should be; the second slider is the target temperature in Kelvins (310 is body temperature); the third slider is friction, which should be lowered once the simulation has started.

In the bottom left is the framerate, and the number of steps of physics that have been evaluated.

The top left displays various information about the simulation. First is the total kinetic energy, though if the program is in the middle of storing outputs, it will display the potential energy (which is what the protein folding process is trying to minimize). Next is the current temperature. Next, "T adjust" displays by what fraction velocities have been rescaled to try to match the target temperature. (Initially, when friction is high, the temperature will be much lower than the target temperature, and "T adjust" will get high.) The next display, "Chunk T" measures temperature via the motion of a large chunk of the protein; if "Chunk T" is significantly higher than the standard temperature measure above, this means that the protein is moving in a more coordinated manner than would be expected at equilibrium, for example if the entire protein is transitioning from a stretched out form to something compact. Finally, "V caps" reports how many times the velocity of an atom has grown so large it has to be artificially capped; if the protein is started in an awkward configuration, this number may jump up during the first few frames, but if it continues increasing, this is a sign of unstable physics, so the timestep should probably be reduced.