BMI 214

Representations and Algorithms for Computational Molecular Biology

Project 3

On this page... (hide)

Due electronically by **11:59PM, Thursday, November 14, 2013**.

Note that programming projects should be completed individually. You may discuss algorithms with others, but the coding should be done alone. Any collaboration must be explicitly mentioned in header comments in your code.

1. Start Early

You should get your code working early for this project, as the actual running of your program can

take quite a while for all required runs!

2. Files to download

Files needed for this assignment are located here:

http://bmi214.stanford.edu/files/p3/download/

3. Introduction

Molecular Dynamics (MD) is using computer to simulate the interaction of atoms and molecules for a period of time under known laws of physics. It was first developed in the area of theoretical physics and traditionally has been applied in material science, biochemistry, and biophysics. Over the last decade or so, molecular dynamics has seen great success in the field of biology, shedding light on the mechanisms by which proteins fold, stabilize, interact, and function. It is now used routinely during structure determination and refinement, and as a tool in drug design. MD is especially powerful in that it allows us to make predictions and engineer drugs that we would not be able to make using sequence or static methods alone. For example, proteins often change conformation upon ligand (the small molecule interacting with it) binding. Given the unbounded protein structure, we may not be able to predict the binding because the functional site is not in the correct shape. Using MD, during the course of simulation, the bounded conformation may appear, and we will recognize the correct binding form.

As in animation, MD consists of many snapshots of atom positions at a series time spots. The principles governing the behavior of molecules are essentially those of chemistry and physics, such as the force fields and Newtonian or quantum mechanics, from which we can derive the interacting forces and compute the positions of atoms at each time spot.

In this project, you will get the chance to implement a molecular dynamics simulation of a small protein. This is going to be fun, because you are making animations of a "dancing" protein!

Start the project early because the simulations take time.

---

4. Overview of a Basic MD Simulation

At its core, an MD simulation can be broken down into simple components. These parts and how they should be implemented within the scope of this project are expanded upon in the following sections.

1. A **biological molecule is represented** within a simulation as a **set of atoms**.
2. These atoms each have mass (though for simplification purposes different atoms may all be given the same constant mass).
3. Some of these atoms are chemically bonded to each other (such as adjacent carbon atoms in a single amino acid), and the others are not (the alpha carbon atoms of different amino acids). Only bonded atoms, and nonbonded atoms within a certain cutoff distance from each other will be considered to interact with each other.
4. Atoms exert forces on one another, depending on whether they are bonded or not, the distances between each pair of atoms, their electostatic charges, and other factors contained within the force/energy model (also called the Force Field) you are using for your simulation.
5. The atoms also start with initial velocities in 3 dimensions, based on the temperature at which the simulation is run.
6. In reality these atoms move smoothly in continuous time; in a model we divide time into little slices (dt), determine the sum of the forces acting on each atom for a given time slice, and use classical mechanics to determine the change in velocity and position of the atom over that time slice.
7. The atoms are then moved, the forces acting on each atom are re-calculated at this new time t+dt, and this is repeated for as many time slices as desired.
8. Smaller time slices are more accurate as we are modeling forces that in reality act continuously and dynamically as atoms move in relation to each other. However, small slices are more costly: if your time slices are 1 picosecond, you need 1000 slices to model motion over 1 nanosecond. Using a 0.1 picosecond slice would more accurately capture the motion of the

atoms, but you would need 10 times the number of iterations. Computation time requirements can build up quickly!

9. The atoms' positions at each time slice can be output to files in specific formats that allow them to be read by various visualization software packages. Displaying atom positions at each time in succession makes a movie showing us how the molecule as a whole behaves as its atoms move during the simulation.

---

5. Project description

5.1 The Simplified Force Field

A given protein structure will have a certain total energy depending on the potential and kinetic energies taken over all of the atoms. As discussed in lecture, the potential energy is calculated using an empirically derived function or "force field" containing several terms corresponding to the different contributions to potential energy: bond, angle, torsion, and nonbonded interactions. Deviations in these terms away from their 'reference' **values will result in higher energy (and thus greater instability).** For this project, we will be using a simplified force field function containing only two terms, one for bond interactions and one for nonbonded interactions (see the lecture notes for a description of the standard 4 term force field).''' Bond interactions are due to direct chemical connections between atoms, and nonbonded interactions occur between atoms with no direct chemical bond.
Our force field for the protein has the following functional form:

$$Energy_p = \sum_{bonds} \left[\frac{1}{2}K_b(b - b_0)^2\right] + \sum_{nonbonds} \left[\frac{1}{2}K_n(r - r_0)^2\right]$$

where,

b_0 = reference bond distance

b = distance between the pair of bonded atoms

r_0 = reference distance

r = distance between the pair of nonbonded atoms

k_b = spring constant for bonds

k_n = spring constant for nonbonds

The first term captures the interaction between chemically connected or bonded atoms, which is modeled as a physical spring (if you remember from physics, the potential energy in a spring is generally E = (1/2)kx^2 where x is the distance from the center of the spring and k is the spring constant). **The first term thus sums the potential energy between all pairs of bonded atoms using the deviation in bond lengths away from their reference lengths.** Note that the potential energy for this term is zero when all bonds are at their reference bond distance. Low or zero energy is favorable for the stability of a structure.

The second term approximates all of the nonbonded interactions in the system and we will also use a spring-type equation to model this (this is a big simplification - see the lecture notes for a more canonical expression modeling nonbonded interactions). **This second term will sum over all pairs of atoms that are NOT bonded to each other, but are within a certain distance from each other.** We will use an input parameter to specify the distance cutoff for nonbonded interactions. In reality, we would need to account for charge, electrostatic and van der Waals interactions, but for the purposes of this project we will be modeling all nonbonded interactions with one simple term.

If two atoms are too far away, the nonbonded interaction between them is negligible. We will have a distance cutoff--if two atoms are not bonded and are closer than the cutoff, then they are considered as nonbonded pair.

5.2 SIMULATING THE MOTIONS OF ATOMS

In order to make the molecule "move", we must know how the positions of the atoms change over time. We get this by (1) calculating the forces on each atom, (2) using these forces to calculate the velocities of each atom, and then (3) applying these velocities to obtain the new position of each atom for each time step.

5.3 Deriving the forces from the force field

In order to compute the acceleration of an atom, we need to compute the total force acting on the atom. To do so, firstly, we need to know how to compute the force acting on an atom from the interaction (bond or nonbond) with one single other atom.

Force is the derivative of the potential energy, and so in order to compute the force, you must derive an expression for force from the terms in the force field. Using the bonded term of our force field as an example:

$$\begin{aligned} PE_{bonds} &= \frac{1}{2}k_b\left(b-b_0\right)^2 \\ F &= \frac{d}{db}PE_{bonds} \\ &= \frac{d}{db}\left(\frac{1}{2}k_b\left(b-b_0\right)^2\right) \\ &= 2\frac{1}{2}k_b\left(b-b_0\right)\frac{d}{db}\left(b-b_0\right) \\ &= k_b\left(b-b_0\right) \end{aligned}$$

This expression represents the force between two atoms paired in a bond, and is proportional in the length of the bond (b) from its reference length (b0). Similarly, you can derive the formula for the force between nonbond atoms. Bond and nonbond lengths are calculated as Euclidean distance between the two atoms in the pair.

The expression for force derived above is actually the magnitude of the force. Force has direction in 3D space. For the convenience in computing the 3D coordinates of an atom in the next time step, we would like to know the force component in each XYZ direction. To do so, we multiply it separately by the proportion of the displacement in each of those directions. For example, **to compute the force acting on atom A from the interaction with atom B**:

$$\begin{aligned} F_x(A,B) &= F\cdot\frac{(x_B-x_A)}{b} \\ F_y(A,B) &= F\cdot\frac{(y_B-y_A)}{b} \\ F_z(A,B) &= F\cdot\frac{(z_B-z_A)}{b} \end{aligned}$$

From the formula, you would notice that, if the distance between two atoms is larger than the reference length, the atom A will be attracted to atom B; otherwise, A will receive a repulsion force. When computing the force acting on atom B from the interaction with atom A, you should switch the A's and B's in the formula. Beware of the signs--you can get very frustrated if you get them wrong!

The total force acting on one atom in each of the XYZ directions is the sum of forces in that direction with all atoms in interaction:

$$
\begin{aligned}
F_x(A) &= \sum_{all\ other\ atoms\ B} F_x(A,B) \\
F_y(A) &= \sum_{all\ other\ atoms\ B} F_y(A,B) \\
F_z(A) &= \sum_{all\ other\ atoms\ B} F_z(A,B)
\end{aligned}
$$

5.4 Updating the positions and velocities

The motion of atoms can be described using mathematical equations incoporating position, velocity, and acceleration, which you may remember from classical physics. However, due to the complexity of the force field, we can only approximate the solutions to these equations numerically using an intergration algorithm. Several integration algorithms have been developed. Some of them are discussed at http://www.ch.embnet.org/MD_tutorial/pages/MD.Part1.html.

In our project, we use the Velocity Verlet algorithm:

$$
\begin{aligned}
v_{t+\frac{dt}{2}} &= v_t + \frac{1}{2}a_t \cdot dt \\
r_{t+dt} &= r_t + v_t \cdot dt + \frac{1}{2}a_t \cdot dt^2 = r_t + v_{t+\frac{dt}{2}} \cdot dt \\
a_{t+dt} &= \frac{1}{m}F_{t+dt} \\
v_{t+dt} &= v_{t+\frac{dt}{2}} + \frac{1}{2}a_{t+dt} \cdot dt
\end{aligned}
$$

where,

r = the position of the atom

v = the velocity of the atom

a = the acceleration of the atom

dt = the length of the timestep

5.5 Keeping track of energy

The total energy of the molecule is a good indicator of the stability of the simulation. It is also a good check on whether your code is working correctly - a proper molecular dynamics simulation with reasonable parameters should have essentially constant total energy (give or take 1 or 2 kiloJoules, in our case). Recall that once the atoms start to move, they will have kinetic energy in addition to the bonded and nonbonded energy. It is normal for KE and PE to trade off throughout the simulation, but if your total energy varies it means something is wrong with your implementation. That means that the total energy

$$E_{tot} = KE + PE_{bonds} + PE_{nonbonds}$$

must remain **constant** - you will be calculating and monitoring all of the energies and the total energy during every time step as part of your project output. **You should keep track of the energy to monitor how your simulation is going and calculate the energy values at EACH time step. Make sure you write to file at every 10th time step (time mod 10). You will need to include all three terms in your total energy calculation and write out all three terms in your output.**

Kinetic energy is a function of velocity and is calculated using the formula:

$$KE = \frac{1}{2} m \cdot v^2$$

You should calculate the kinetic energy after updating the velocity at t+dt ('**not the half time velocity**'). The total kinetic energy is also the sum of the individual total kinetic energies computed from the velocities in each of the three dimensions. Note that force is a vector, but energy is a scalar. You do not need to consider kinetic energy when calculating the forces on each atom.

Note that the pairs of nonbonded atoms technically may change between time steps as a result of atoms moving within or beyond the distance cutoff. For our purposes, you should only find the nonbonded pairs once - at the beginning of the simulation using the input files - and use these as a constant set throughout the simulation.

For each pair of interacting atoms, you only need to compute the bond or nonbond energy ONCE per iteration.

5.6 Calculating velocities from temperature

In order for you to understand the effect of the temperature parameter in simulations, we have created three different versions of the input file that will run the same default simulation, but with different initial velocities corresponding to the different temperatures. The temperature of the simulation primarily affects the initial distribution of atomic velocities. This is discussed at http://www.ch.embnet.org/MD_tutorial/pages/MD.Part1.html#Verlet (and other places).

5.7 SUMMARY OF STEPS FOR CODING THE SIMULATION

1. Parse an input file containing atom, coordinate, initial velocity, connectivity, and parameter information.
2. Iterate through 1000 time steps, and for each time step do the following:
    - Update the velocities (for t+dt/2) on each atom (the forces should be initialized to zero when you do this in the first time step, so your velocities won't change yet)
    - Update the positions of each atom
    - If you are keeping global variables for forces and energy then make sure that they are reset to zero.
    - For each interaction pair (bonded and nonbonded):
        - Calculate the potential energy and update the total potential energy
        - Calculate the force and update the total forces in each dimension for each atom
    - Update the velocities (for t+dt) on each atom and calculate the kinetic energy
3. Write the appropriate output every 10 time steps. Do this for all file types: (rvc, crd, files needed to generate euc)

---

6. Data input and output

6.1 Data Input

Input to your program will consist of an input filename and a set of optional parameters to replace default values. Your program MUST be able to accept these options on the command line. The parameters options your program should accept are:

parameter	description	default value
--if	input filename	no default value, must be specified
--kB	same as kb	40000.0

--kN	same as kn	400.0
--nbCutoff	distance within which atoms should be considered as having nonbonded interactions, if they are not a bonded pair	0.50
--m	atom mass, a constant to be applied to all atoms	12.0
--dt	length of time step	0.001
--n	number of time steps to iterate, useful for debugging	1000
--out	prefix of the filename to output (for grading purposes). For example, if --out is testRun10, the outputs would be testRun10_out.erg and testRun10_out.rvc (see Data Output section for more details) string passed into	--if (stripped of the suffix .rvc)

An example of a call to your program replacing the default kb value with a value of 800.0 is:

```
python MyMD.py --if ./input.rvc --kB 800.0
```

The input files (RVC files) are of extension .rvc (r = positions, v = velocities, c = connectivity), specify the positions and velocities of every atom in the simulation, and contain one line per atom. The first line in the file is always a header line containing the parameters used to generate that file (all will be the default values, except for the temperature). Each line after the first consists of (in white-space-delimited format) the atom ID number (1 indexed), the atom's starting x coordinate, y coordinate, z coordinate, initial x, y, and z velocities, and then up to 4 atom ID numbers corresponding to the atoms "connected", or bonded, to that atom.

The input files essentially represent frame 0 of the simulation; all values are at time t = 0. You will calculate reference lengths and other reference values using the input file. You will use the connectivity information to determine the bonded pairs of atoms, and you will need to calculate inter-atom distances from the initial configuration and compare to nbCutoff to find all the nonbonded pairs of atoms. These nonbond reference distances (r0) are fixed through the simulation, and they do not change as the molecule moves, since they represent the "forces" that hold together the initial structural configuration. You should also calculate your reference bond lengths from the input file. Note that we are using a constant mass across all atom types (this is a simplification - in reality, every type of atom has a different atomic mass).

For many of the simulations you will be doing, however, you will only be changing one or two parameters from the default, so it is best to hard code the default parameter values and override these if command line options are used. You should also run each simulation for 1000 time steps, so you can hard code this value in as well, though if you are inclined to make longer simulations (not required for this project) you may modify this. Even though you will only be doing 1000 timestep simulations in this project, ensure that your code handles the --n flag (controlling the number of timesteps) correctly; it is required!

The input files you will be using are: 1FW4_cold.rvc, 1FW4_med.rvc, 1FW4_hot.rvc, and 1FW4_noCa_med.rvc These input files were generated using different temperatures (50K, 350K, and 4500K) and so have different initial velocities. One (1FW4_noCa_med.rvc) does not include the two calcium atoms.

6.2 Data Output

The output of your program will be two different files, a .erg (ERG) file and a .rvc (RVC) file.

The ERG file should contain your energy values for each time step, and should look like the example (with different values, of course). The file supplied icludes all 1000, whereas you should be supplying every time mod 10. The lines should be tab-delimited, with the time step number in the first column, the kinetic energy in the second column, the bond potential energy in the third column, the nonbond potential energy in the fourth column, and the total energy in the last column. All values should be written out to exactly 1 decimal place.

The RVC output file will look very similar to the input RVC file, but is tab-delimited. During simulation, you should output all the atoms' IDs, positions, velocities and connectivities every 10

time steps, concatenated together into one RVC output file, like the example. You should include the original input RVC file and you should use 1000 as the default number of total steps (feel free to perform longer simulations if you have the time and inclination), so you should have about 100 frames in your output file.

Like the example, each frame should be separated by a '#' comment so they can be easily parsed as separate frames. It is helpful if you also include the frame number and energy.

You should then use the following Python script to convert your RVC files into AMBER coordinate files (.crd), which will be needed to visualize your simulations: convertRVCtoCRD.py

To run this script on a machine with Python installed, simply type:

```
python convertRVCtoCRD.py input.rvc output.crd
```

AMBER stands for "Assisted Model Building with Energy Refinement", and refers both to a set of force fields commonly used in MD simulations of biomolecules, as well as to a popular package of MD simulation programs. The visualization program we are using is capable of loading the AMBER file format.

You will be running your simulation code on a fragment of calmodulin, a calcium binding molecule involved in a very diverse set important signalling processes such as muscle contraction the creation of memories. We have included two different versions of a modified PDB file, which has fewer amino acids and has the hydrogen atoms removed to reduce the number of calculations needed for your simulations. The file 1FW_abbreviated_renum.pdb is the pdb file that contains a representation of the molecule including two bound calcium. This is the file which you should use on most of your visualization runs. The file 1FW_abbreviated_noCa_renum.pdb does not include the two calcium and can be used to visualize simulation runs which don't include the calcium.

Note that in simplified description of force, we will consider the calcium atoms to be 'unbound' to any atom in the protein. Our model includes an energy term that involves interactions between unbound atoms. Do you think that this is a sufficient description of the coordination between the side chains in the binding sites with the calcium ions? How do you think it compares to other intra-molecular forces between non-covalently bound atoms in the protein?

Perform these 11 simulations. Your output MUST be to files within the same directory where your script is run.:

Type of Run	Input File	Parameter Settings	Output Files
Default	1FW4_med.rvc	defaults	1FW4_med_out.rvc, 1FW4_med_out.features, 1FW4_med_out.crd, 1FW4_med_out.erg
Low Temp	1FW4_cold.rvc	defaults	1FW4_cold_out.rvc, 1FW4_cold_out.crd, 1FW4_cold_out.erg
High Temp	1FW4_hot.rvc	defaults	1FW4_hot_out.rvc, 1FW4_hot_out.crd, 1FW4_hot_out.erg, 1FW4_hot_out.features
Weak Interactions	1FW4_med.rvc	kB=1000.0, kN=100.0	kB1000kN100_out.rvc, kB1000kN100_out.crd, kB1000kN100_out.erg
Near Interactions	1FW4_med.rvc	nbCutoff=0.25	nb25_out.rvc, nb25_out.crd, nb25_out.erg
Far Interactions	1FW4_med.rvc	nbCutoff=0.75	nb75_out.rvc, nb75_out.crd, nb75_out.erg
Longer Time Steps	1FW4_med.rvc	dt=0.01	t01_out.rvc, t01_out.crd, t01_out.erg
Much Longer Time	1FW4_med.rvc	dt=0.05	t05_out.rvc, t05_out.crd, t05_out.erg
Very Long Time Steps	1FW4_med.rvc	dt=.1	t1_out.rvc, t1_out.crd, t1_out.erg

No Calcium	1FW4_noCa_med.rvc	defaults	1FW4_noCa_med_out.rvc, 1FW4_noCa_med_out.features, 1FW4_noCa_med_out.crd, 1FW4_noCa_med_out.erg
Mutated	1FW4_MUT_med.rvc	defaults	1FW4_MUT_med_out.rvc, 1FW4_MUT_med_out.crd,1FW4_MUT_med_out.erg

6.3 Notes

1. **You must also create the files sitesCa1.euc and sitesCa2.euc. These will be explained in the next section.**
2. Be aware that some of these parameter sets (certainly while varying dt) will result in very large values and may cause overflow errors. It is a good idea to catch those errors when they occur. This is one manifestation of the simulation becoming "unstable" - i.e. the molecule is behaving outside its physical bounds. We will be asking you to determine the time step value at which this seems to occur. **A good way to check this is to see if the total energy increases or decreases by a factor of 10 compared to the initial, starting, energy.**

6.4 COMMON BUGS AND FIXES

- **Applying force in the wrong direction for each pair of atoms.** This means that you are adding force to an atom when you should be subtracting it, and vice versa. A result of this is increasing energy. To fix, just switch the operation.
- **Counting your bonded pairs as nonbonded pairs.** You should only consider a pair of atoms to be nonbonded if the initial distance between them is below the nbCutoff AND if the atoms are NOT BONDED. This will also result in increasing energy. Just make sure that you check whether two atoms are bondeds (using the connectivity information in the input RVC files) before labeling them as a nonbonded pair.
- **Forgetting to zero force and energy before updating them.** You must set all forces and energies to zero before calculating them during each time step. If you don't, your energies will obviously increase.
- **Poorly optimized code...** If your code takes forever (> 5 minutes) to complete one 1000 time step simulation, you should think carefully about your data structures and implementation, and perhaps your choice of programming language. See the section on code optimization.
- **Double counting bonded and nonbonded energy.** A bond or a non-bonded interaction should only contribute to the total bonded / nonbonded energy once.
- **Applying the wrong order of operations.** Make sure you followed the Velocity Verlet algorithm.

---

7. Data Output: Origins of stability

In your various runs from the previous section, we performed three similar experiments:

1. **The Default protein at normal temperature, bound to calcium.**
2. **The Default protein at normal temperature, not bound to calcium.**
3. **The Default protein at normal temperature, mutated sites near calcium.**

Since we are keeping temperature constant, we can ask the following questions:

- Does the presence of calcium affect the stability of atoms with respect to each other?
- Does the mutation of atoms (and thus changing how they interact with other atoms) affect the stability of atoms with respect to each other?

You will be asked to address these questions in the quiz. We will explore two pairs of atoms associated with calcium binding in order to determine the effect of calcium binding as well as local mutations. For the three runs listed above, create two tab-delimited files called sitesCa1.euc and sitesCa2.euc, both in the following format (these values are arbitrary):

```
0       0.4730  0.4730  0.4730
10      0.4721  0.4721  0.4721
20      0.4711  0.4711  0.4711
```

```
30        0.4702   0.4702   0.4702
```

- The first column is the time step (up to 1000)
- The other three columns each correspond to the three runs listed above

Each of these last three columns is a calculation of the distance between two atoms associated with a calcium binding site. **For sitesCa1.euc, each of these columns is the Euclidean distance between atoms 17 and 96. For sitesCa2.euc, each column is the Euclidean distance between atoms 298 and 385.**

Submit 2 graphs of these (line graph: y-axis is the Euclidean distance, x-axis is the time step). Also include a legend identifying the three different lines. These two graphs should be in a format such as pdf, gif, tiff, or jpeg.

---

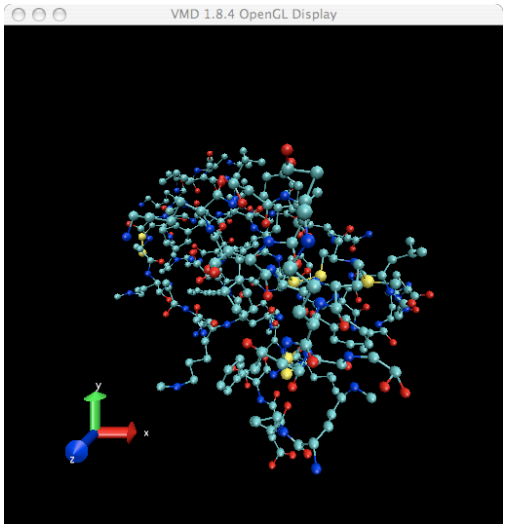
8. Visualizing MD simulations

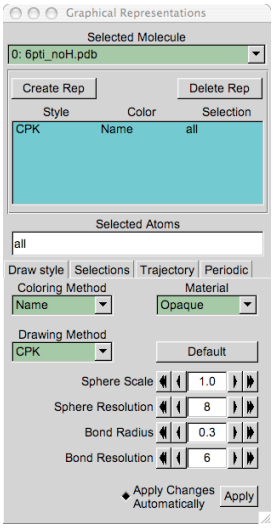
8.1 VMD

The most interesting part of MD is arguably the visualization. Humans are very visual, and being able to see how the molecule moves and behaves allows us to understand and reason about the mechanisms underlying the molecule's behavior, whether it be as complicated as a protein recognition and binding event, or as simple as a molecule oscillating around a stable conformation (i.e. "breathing") under normal conditions. Typically, a simulation is saved as a number of frames, a frame being a recording of all the atom positions at a particular time point. Visualization software then "plays" each frame of the molecule in quick succession, just like an animation. What you can then see is how different properties of the molecule change over time - bond lengths, motion of amino acid side chains, speed and displacement of atoms from each other, overall molecule structure, etc. We will be asking you to view your simulation movies and answer questions about them in the project quiz.

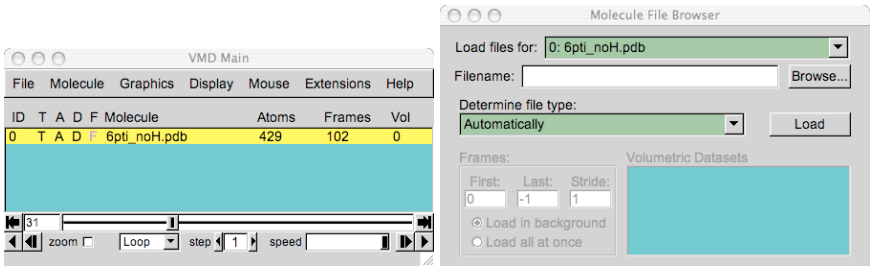
There are a number of visualization packages available for viewing MD simulations; one of the most popular is the free software VMD. VMD works on almost all platforms, including Windows XP, MacOS X (10.3.5 or later), and Linux. The brief guide to VMD that follows is based on the Mac version of the software.

Download VMD (registration required), and follow the installation instructions.





VMD Display window Graphical Representations window



VMD Main and Molecule Browser windows

8.2 To view one simulation:

1. Open VMD

- You should see a "Display" window and a "Main" window.

1. Load the PDB file for the molecule

- In the "Main" window, go to "File" and then "New Molecule..." This brings up a Molecule File Browser. Select "Browse..." select your PDB file (be sure to use the noCa file only with the simulations that were generated without calcium, otherwise you will get a mess) and click "Load"

1. Load the simulation file

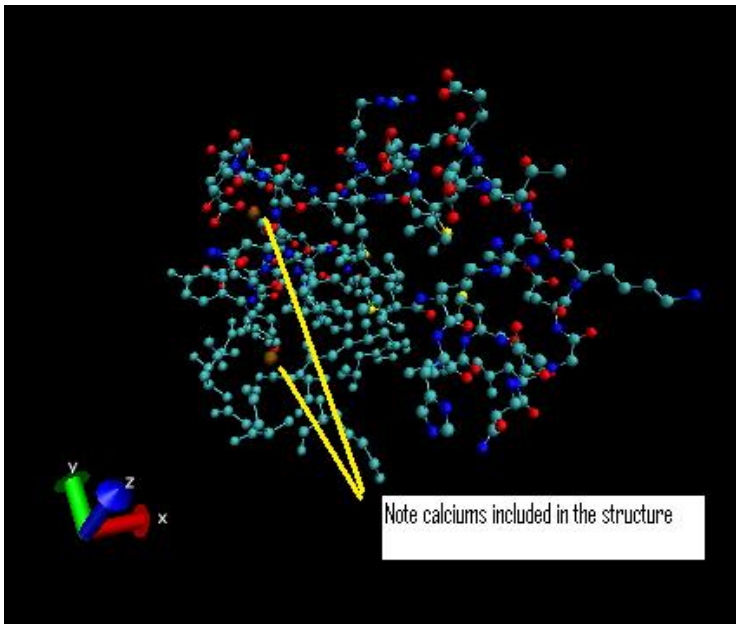
- In the Molecule File Browser, click "Browse..." again and select your simulation file (extension .crd, more about this format in the data output section)

1. Play the simulation

- You can adjust the speed of the simulation using the speed slider bar at the bottom right of the "Main" window
- You can also rotate and zoom the simulation using your mouse

1. Try different ways of drawing the molecule

- In the "Main" window, go to "Graphics" and then "Representations..." This will bring up a "Representations" window (and the "Draw style" tab should be active). Choose drawing methods from the "Drawing Method" drop down menu (default is "Lines"). DON'T CHOOSE 'SURF' OR 'ISOSURFACE', AS THESE WILL TAKE TOO LONG TO RENDER. Good ones to choose are "Cartoon" or "New Cartoon" (to see secondary structure only), or "CPK" (atoms drawn as spheres, bonds as lines). "CPK" is probably the best representation for seeing details like bond length changes and side chain behavior. You will also need to use a method like "CPK" or "VDW" to see the calcium atoms. You should also choose "Element" from the "Coloring Method" to highlight the different element types. To see the sites of calcium binding, it may help you to orient the three axes as shown below.



Visualization of calmodulin with calcium.

To load a new view, select the molecule in the "Main" window and either right click on it or go to the "Molecule" menu. Choose "Delete Molecule" and click OK. Then load a molecule and simulation file as before.

You can view multiple simulations (for comparing/contrasting) by opening multiple instances of VMD. You can also load multiple molecules and attach different simulations to them in the same window, but this is not recommended. We have supplied you with two pdb files of this specific section of calmodulin, one with calcium and one without. We do not supply a mutated pdb file, and you will not be asked to visualize it.

---

9. Molecular Interactions

Protein-ligand interactions are extremely important in variety of circumstances, particularly in understanding pharmacological activity. Very slight nuanced differences in how a small molecule interacts with a protein domain such as an enzyme's site of activity, a receptor binding site, or an ion channel can have profound differences in activity. It can be the difference between activation and inhibition and is at the core of drug side effects and why people can respond differently to exactly the same drug. You can find out more at PharmGKB.

To relate this to our investigation of structurally derived features (FEATURE), you can use your simulation runs to evaluate changes in the scores for the binding sites as they evolve over time. We will investigate whether or not the presence of calcium in the binding site helps stabilize that site over time and if it effects the evolution of the score of the site. As mentioned, we have provided two different versions of the calmodulin fragment. One includes a calcium ion in each of the two binding sites, the other does not.

Keep in mind that we are cutting several corners here, as electrostatic forces between a calcium ion and the charged and non-charged atoms in a molecule are very different, and we are computing scores that were derived from a very impoverished feature set with limited examples. We also will be tracking the feature scores at fixed points as the simulation progresses rather than calculating a position relative to the changing atoms in the molecule or over a whole grid of the molecule. However, you can hopefully get a taste of how these sorts of simulations can be used to investigate molecular interactions.

We have included a python script to calculate the calcium binding site scores for this calmodulin fragment. To run this script on a machine with Python installed, simply type:

```
python computeFEATUREScore1FW4.py input.rvc output.features
```

You should create a graph (using graphing or spreadsheet software of your choice) showing a comparison of the calcium binding site scores using 1FW4_med_out.features and 1FW4_noCa_med_out.features which you can extract using the script on the appropriate out.rvc files. Your graph should have time steps along the horizontal axis and scores along the vertical axis with four different lines (connected set of points), with a figure title and a legend identifying the four different lines. It should be in a format such as pdf, gif, tiff, or jpeg. Then you should create another graph comparing 1FW4_med_out.features and 1FW4_hot_out.features.

---

10. Code Optimization

Due to the large number of operations required in a simulation, sub-optimal implementation may result in extremely long running times. There are a number of ways to increase efficiency and improve running time:

1. **Avoid unnecessary looping**
    - There are many loops over all atoms required during each time step, but you can minimize the number of loops made
2. **Use smart data structures**
    - There are ways of storing the data that can improve the efficiency of look up and even the efficiency of looping. You might not have to loop over all atoms to compute energy and force, for example.

---

11. Deliverables

11.1 Quiz

You will run your code to answer the questions in the quiz:

Project 3 Quiz

Use this template to fill in your answers:

project3_quiz.txt

11.2 Details for submission

1. Commented, working code with a README.txt containing instructions for running your code. Name your main script with the following format: md_sunetid.py (or whatever extension specific to the language you are using)
2. 11 .rvc, .crd, and .erg files + 3 .features files corresponding to the simulations we specified above.
3. 2 .euc files + 2 line graphs; both depicting the changes in distances between two atoms in varying conditions. Please describe how you created these files in your README.
4. Two graphs showing the calcium binding site scores of the two sites in the presence/absence of calcium and at medium/high temperatures as .pdf, .gif, .jpg, or .tiff
5. Completed project quiz.
    - Use the project3_quiz.txt template file to fill in your answers.
    - Make sure you include at the top of your quiz: your name, sunetid. This is so when it is printed out, it is clear who it belongs to.
    - **Your code should not include any non-standard (ie not included with the basic installation) libraries**. Specifically, it should be able to be run from the Stanford corn machines (Note: NumPy is now installed on the corn machines as default with Python).

---

12. Submission instructions

1. Log in to a corn (or tree, cardinal, etc.) machine:
   -> `ssh sunetid@corn.stanford.edu`

2. Place all of your files from 1-5 above into one directory, e.g.
   -> `~/biomedin214/p3/submit`
3. Please only include those listed above. Delete executable binaries if you have them (e.g. if you used Java/C/C++); we will compile your source code.
4. cd into your submit folder:
   -> `cd ~/biomedin214/p3/submit`
5. Run the class submission script:
   -> `/usr/class/biomedin214/bin/submit p3`
   Note the space before p3. Be sure to run the script from within your submission folder (where all your files to be submitted are)!

You may submit multiple times; simply re-run the script. Each new submission overwrites the previous submissions. Your submission date will be the final submission received, and late days will be charged accordingly. **Don't forget the quiz!**

Page last modified on October 30, 2013, at 01:32 PM