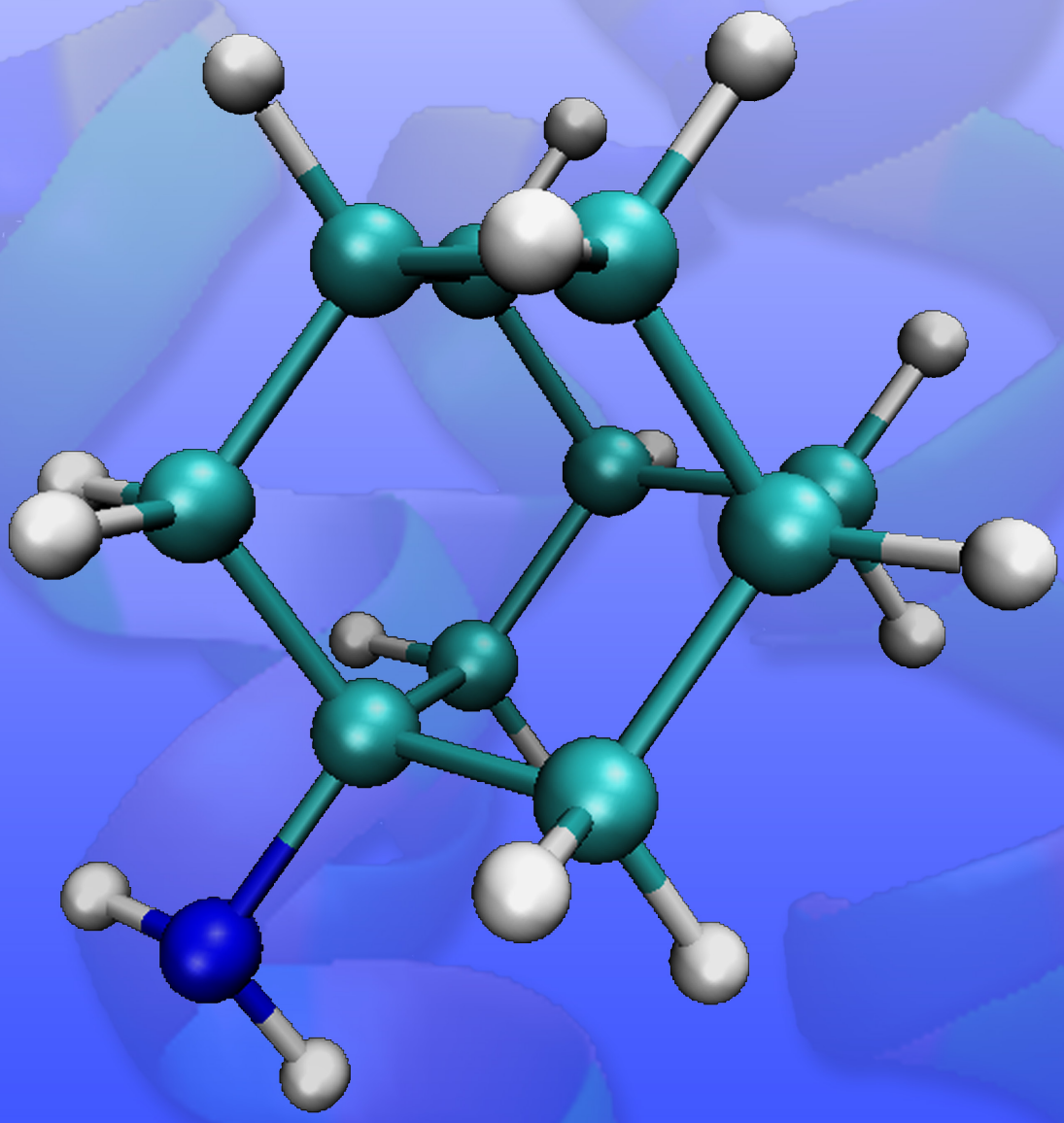
PDBIO 550R

Molecular Modeling



Busath Lab

Contributors:

- David Busath: Labs 1-4, 6
- Richard Swensen: Various
- Mitchell Gleed: Lab 8, lab manual
- Curtis Evans: Labs 7, 8
- Mark Fowler: Lab 7
- Matt Durrant: Lab 5

Last modified on April 15, 2016 9:06 PM

TABLE OF CONTENTS

LAB 1: INTRODUCTION TO THE PROTEIN DATA BANK AND VMD

In this lab you will be introduced to two important resources vital to molecular modeling. The first is the Protein Data Bank website. And the second is a program called VMD. By the end of this lab you should be able to use the PDB website to find a protein in the database and download the protein's .pdb file to your hard drive. You should also become familiar enough with VMD so that it can be used as a tool to understand what is going on in you modeling simulations.

I. RCSB Protein Data bank

Many of your molecular models will contain proteins, many of which are common in dynamics simulations. It would be very time consuming to have to recreate these proteins every time that you wanted to use one. The **RCSB Protein Data bank** contains a database of proteins that you can use in your models.

Finding proteins in the database

To get started, use your internet browser to go to the RSCB homepage. The website is located at pdb.org. We will use the Botulinum Neurotoxin, often referred to as BoNT or BoTox, in this first lab. If you knew the PDB ID number for your protein, you could simply type it into the search box and click Search. Since you probably don't, you can just type BoTox in the search box and click Search. The search should bring a list of results. We want to use the crystal structure of Botulinum Neurotoxin Serotype A, which was the first one determined and was released in 1999. The four digit code on the left hand side above one of the small pictures is the **PDB ID** number. We are looking for 3BTA. Click on it to bring up a page specific to our protein. This page contains a lot of information that we will not cover in this lab.

Downloading PDB files

You can download files from the PDB website to your computer so that you can use them in simulations. On the page specific to our BoTox protein, toggle the drop-down menu on the right-hand side labeled "Download files" and choose "PDB Format." Save the file where you will be able to find it later.

II. VMD

We will be using a program named **CHARMM** (Chemistry at HARvard Molecular Modeling) to run our simulations. While CHARMM is great at crunching numbers to simulate how systems work, it has no graphical component. **VMD** is a program that converts the output files from CHARMM to visual representations of molecular system as if we were looking at them under some type of microscope.

Loading files into VMD

When VMD is opened, it starts three windows: a console window, a display window, and the VMD main window. On the main window, click "file" then "New Molecule…" This brings up the "Molecule File Browser" window. Make sure that the drop down box next to "Load files for:" says "New Molecule." If it does not, arrow select "New Molecule." Next, click the button that says "Browse…" A new window will appear. Find the file that we downloaded from the PDB website hopefully named "3BTA.pdb"

and click "Open." The "Molecule File Browser" window should now have the path of the file you just selected in the white text box. Click the button that says "Load" and close the window. In the Main window, you should see a line of text with several numbers and the filename that you just opened. The display window should have a picture of the protein encoded in the file you just opened.

Display options

Rotation: The rotate mode allows you to rotate the display. In the main window, click on the "Mouse" pull-down menu and select "Rotate Mode." Next click and drag anywhere in the display window. If you click, drag, and then let go of the mouse button while the mouse is still moving, VMD will continue to rotate or spin the display even though you are not holding down the mouse button. If you click, drag, and then let go of the mouse button when the mouse is not moving, VMD will not make the display spin. Try clicking and dragging with the right mouse button and see what happens. The display should rotate differently when holding the right mouse button. It will not be a free rotation like you observed when you held the left button.

Scaling: The scale mode allows you to zoom in on the display. Again this mode is activated by selecting "Scale Mode" in the "Mouse" pull-down menu. To zoom in, click and drag on the display window. Drag left to zoom out and drag right to zoom in. If your mouse has a scroll wheel, then you can also just spin the wheel to zoom in or out.

Translating: The translate mode allows you to move the display. To enter this mode, select "Translate" from the "Mouse" pull-down menu. To move the display, simply click on the display window and drag it the directions you want.

Labeling: The label mode allows you to show the names of different components of the display. This mode is also found in the "Mouse" pull-down menu. You have four options. You can label atoms, bonds, angles, and dihedrals. To label an atom, just click on it. To label a bond, click on the two atoms in the bond. To label an angle, click on three atoms. To label a dihedral angle, click on the four atoms that comprise the dihedral. *To manage labels, use the "Labels" tool under "Graphics."*

Reset: Now that you have rotated, moved, and zoomed in and out on you display, it might look a little funny. You can reset it by selecting "Reset View" from the "Display" pull-down menu in the main window.

Perspective vs. Orthographic: Perhaps you noticed that when you rotated the display, some bond lengths seemed to get smaller, and some got bigger--This is the result of using **perspective mode** in VMD. In an attempt to help visualize the three-dimensional structure, VMD makes atoms that are far away small and atoms that are close big. Often times this feature makes it hard to see what is going on in the system. To turn off perspective and enable **orthographic mode**, select "Orthographic" from the "Display" pull-down menu in the main window.

Stereo Vision: Another tool to help visualize the three-dimensional structure of your system is **stereo vision**. In the Main window, select "Display," "Stereo," and then "Side by Side." The display window should have two images side by side. This function is like magic eye pictures. The idea is that you want your right eye to look at the image on the right and your left eye to look at the image on the left. The images look the same, but are slightly different. When the two images are superimposed in your brain, you get the illusion of 3-D, or "stereo vision."

This can be a little tricky at first, so here are some pointers. First, resize your window so that the centers of each image are about two inches apart. Next, get as close to the screen as possible without touching your nose to the glass. The idea is that if you are close enough, then your right eye can only see the right image because your nose blocks the left image. Your left eye likewise can only see the left image. Relax your eyes as if you were looking across the room. You should see one unfocused image. Your brain will think that the two different images are really the same image. Slowly back away from the screen being careful to focus so that the single image remains super-imposed. As you back away, your right eye will begin to see the image on the left and your left eye will be able to see the image on the right. You will see a total of three images. The one in the middle will be in 3-D! Sometimes it helps to slightly rotate the display with the mouse. You can also zoom in now (scale up), but not too far or you will lose focus.

Graphical Representations

You can change the way different atoms and bonds are represented by using the Graphical Representations window. To open this window, use the "Graphics" pull-down menu in the Main window and select "Representations…"

A **representation** is a group of atoms and its display properties. You can create and delete representations with the "Create Rep" and "Delete Rep" buttons. As you do, they will appear and disappear from the text area below the buttons. If you double click on the representation in the text area it will toggle between active and inactive. When the representation is active, it shows up in the display window. When it is inactive, it disappears.

Draw style tab: This tab allows you to change how each atom and bond will be represented on the screen. The "Coloring Method" refers to the way VMD determines how to color the atoms and bonds. You choose how to group the selected atoms for the representation. For example, you could choose to color the atoms by name. All of the carbon atoms would be one color and all of the hydrogen atoms would be another color. You could group atoms by molecules. All of the water molecules would be one color and all of the lipid molecules would be another color. The "Drawing Method" changes the shape of the representations. Line representation just draws lines to represent each bond. The color of the line is determined by the coloring method. The "VDW" method draws each atom as a small sphere. The "Cartoon" method shows structural components of proteins like alpha helices and beta sheets.

Selections tab: This tab allows you to select which atoms are in the representation. The "Singlewords" box allows you to choose common groups of atoms such as all, water, protein, etc. The "Keyword" box gives you a list of identifiers defined in the pdb file. We will discuss many of these identifiers in later labs. For now you should know that the "name" keyword refers to the name of the atom. The atom names in the .pdb file are not necessarily the simple element names learned in chemistry. If you click on name, a list of all the atom names in the file will appear in the "Value" box. To make a representation that selects only carbon atoms first clear the text box below where it says "Selected Atoms." Next, double-click on the keyword "name." You will notice that there are many different names that represent carbon. Double-click on the "C" located in the value box. To save the selection, click "Apply."

Other Tools

Clipping Planes: If you want to see a cross-sectional view of a system, you can use the **clipping plane tool**. To do this, use the "Extensions" pull-down menu. Select "Visualization" then "Clipping Plane Tool." You can have up to 6 clipping planes. To edit any one of the planes, click on the button with its number. You can tell if a particular plane is active in two ways. The first is that there is a small check-box by the title. You can also tell because when a plane is active there is a white bar under its button. Click on clipping plane number '0.' When you first turn on a clipping plane, the plane is parallel to the screen. All of the atoms behind the plane disappear. If you click on the "flip" button, then all of the atoms that were hidden appear and the atoms that were visible are hidden. The clipping plane is perpendicular to a vector. You can manipulate the clipping plane by changing the three parameters of the vector; the origin of the vector "Origin," the magnitude of XYZ components "Normal," and the distance from the origin of the vector "Distance." The setting "Normal follows view" sets the Normal to the visual plane and allows the molecule to rotate through this plane. The setting "Origin follows center" sets the origin of the vector to the center of the molecule. The default setting sets the origin of the vector at the center of the molecule and you can adjust the distance that the plane is from the origin with the "Distance" sliding box. Move the distance box left and right. The plane will move backwards and forwards.

Rendering: Often you want to use an image from VMD in an outside project such as a poster or a paper. You can export snapshots of VMD using the **render** command. These files are saved in the .bmp format. To save a picture of the display window, use the "File" pull-down menu and select "Render." In the Render Controls window, choose a location to save the snapshot.

Assignment 1: open a copy of the BoTox molecule. Rotate it, rescale it, and translate it slightly. Next, label an atom, a bond, an angle, and a dihedral angle. Finally, render a copy of your display naming it "lab1_singleBoTox.bmp." Save the file to submit it at the end of the lab. Your display should resemble Figure 1 below.

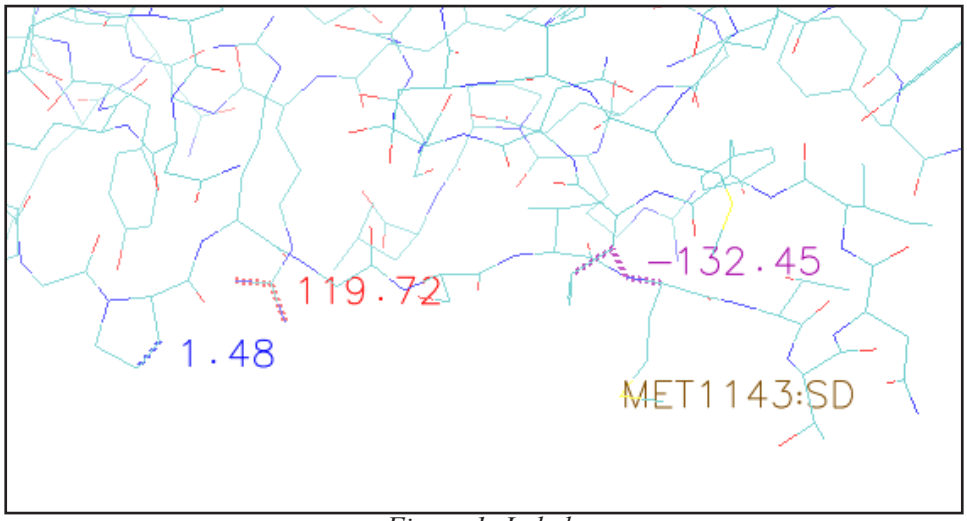


Figure 1: Labels

Multiple files

VMD is not limited to opening one file at a time. You can open many files and view them at the same time. You can even open one file many times, and VMD will load several copies of that molecule. When opening a new file, make sure that the file browser window says "New Molecule" next to the

words "Load files for." Load the BoTox file several times. You know that it is loading because in the main menu there are several lines each with a unique ID number.

When you look at the display window, it looks like there is only one molecule. This is because they are all loaded on top of each other. Try to move them using translate mode. You will notice that they all move together. The same is true in rotate mode and scale mode. To move each copy individually, use the "Move" command found in the "Mouse" pull-down menu. This command allows you to select and move individual groups of atoms. Select "Move Molecule" and drag one of the BoTox molecules off of the stack. To change the representations of any individual file, go to the graphical representation window and select the file name using the right arrow in the box under the heading "Selected Molecule."

Assignment 2: open four copies of the BoTox molecule. Move and scale them so that all four molecules are visible and so that none are touching. Change the representations so that all four molecules have different coloring methods and different drawing methods. Render a copy of your display naming it "lab1_fourBoTox.bmp." Save the file to submit it at the end of the lab. Your display should resemble Figure 2 below.

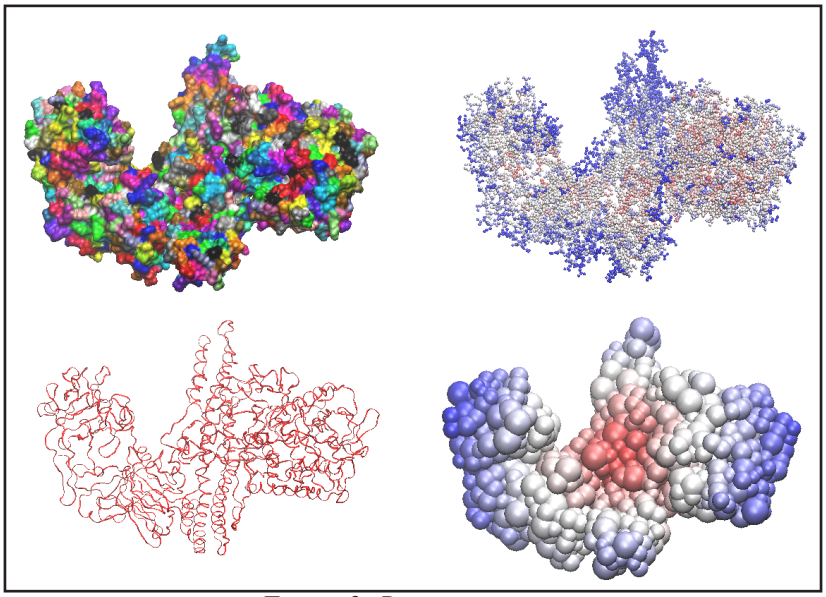


Figure 2: Representations

DCD files

Up until now we have learned how VMD allows us to see molecular models graphically. We have learned how to manipulate the way we look at these models, but the models remain unchanged. We can also use VMD to watch simulations. During dynamics simulations, CHARMM creates **.dcd files** that can be viewed with VMD much like DVD's are viewed with a DVD player. To do this, you need to load at least two files. The first file is the **.pdb file** for the model. The .pdb file has all of the atom information including initial coordinates and atom names for each atom. The other is a .dcd file which tells how each atom moved during the simulation. You can actually load many .dcd files at once.

For this part of the lab, we will watch a lipid bilayer simulation using files in the lab directory. First load the file named "bilayer.pdb" into VMD. When this file loads, you will be looking at the system, which consists of two types of lipid molecules sandwiched by water molecules, from the top. Rotate the system so that you are looking at it from the side. You should see red and white water molecules on

the top and bottom of the system. Next, go to the Main window and select "Load Data into Molecule" from the "File" pull-down menu. You should see the Molecule File Browser window appear just like when you were loading a new .pdb file. Make sure that the Load files for box says "bilayer.pdb." If it does not, then click on the arrow on the right of the box and select "bilayer.pdb." Now browse, select, and load "bilayer.dcd." You will notice that the number of frames in the Main window will start counting, and the display window will start showing the simulation. When the file is done loading, then the frame count will stop. The controls at the bottom of the main window control the simulation playback. The single arrows on the bottom right and bottom left corners play the simulation forwards and reverse respectively. Click on the play button again and the simulation playback will be paused. The arrows directly above the play buttons jump to the beginning or to the end. The last two arrows step through the simulation one frame at a time.

In order to keep system size small while simulating large systems, CHARMM uses something called **periodic boundaries**. Simply, copies of the system are replicated about the original system in all directions. When you load a .dcd file, VMD only shows the primary image. To see the images in the periodic boundaries, defined in the header of the .dcd file, you have to turn them on. To do this go to the Graphical Representations window, select the "Periodic" tab, and click on the check boxes next to the dimensions that you want to show.

Frame 0 of the trajectory is simply the coordinates of the original pdb file, and it won't demonstrate periodicity. Delete frame 0 to show only the frames from the dcd file.

Assignment 3: open the .pdb and the .dcd files for the lipid bilayer. Rotate the display so that you are looking at the side of the model. Make three representations with the following selections: "water", "resname PALM or resname PCGL", and "resname DLPC". Change the drawing method of the representation with the 'resname DLPC' selection so that it is easily distinguishable from the representation with the "resname PALM and resname PCGL" selection. Turn on the periodicity for all the representations except the water representation in both the +x and –x directions. Render three snapshots: the first at the beginning of the run, the second in the middle of the run, and the last at the end of the run. Name these files "lab1_beg.bmp", "lab1_mid.bmp", and "lab1_end.bmp". Your display should resemble Figure 3 below.

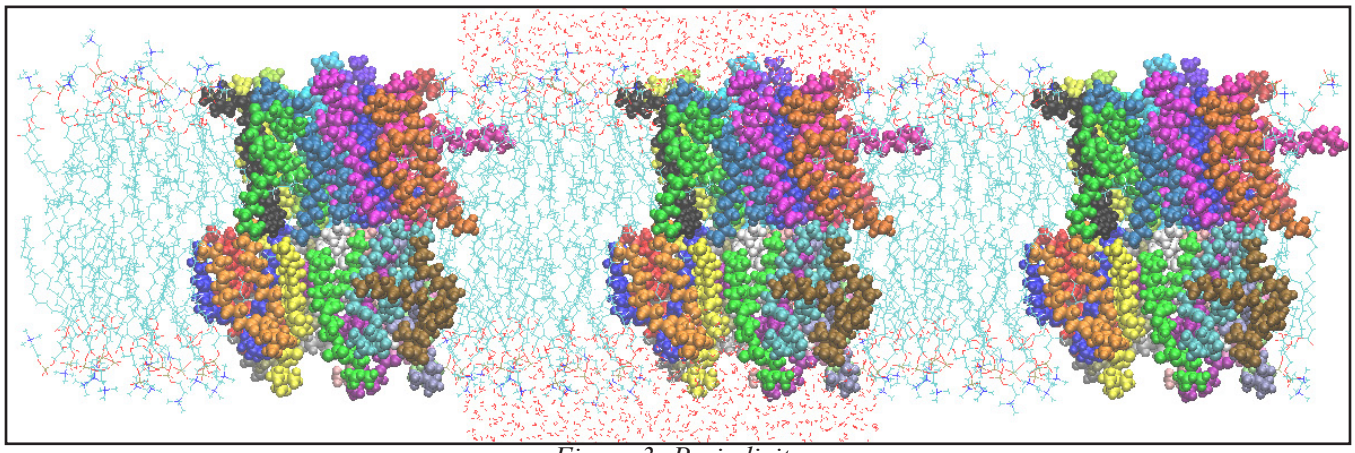


Figure 3: Periodicity

Submit these three files along with the two files from the first two assignments in the lab to your T.A.

LAB 2: INTRODUCTION TO CHARMM

In this lab we hope to help you get comfortable with the rudiments of CHARMM.

I. Linux Basics

Molecular modeling dynamics simulations can require very large amounts of computational power. Because of this, CHARMM was written with the capability to use more than one processor at a time. This is part of the reason why CHARMM was written to run on the Unix or Linux operating system instead of the Windows operating system. For those of you that are not familiar with Linux and Linux text editors, this section should give you a crash-course that will enable you to complete your labs.

In this lab, we will actually be using a small Linux cluster, named **Wolf**, which is located in Dr. Busath's office. Fortunately, we can all simultaneously log onto Wolf using a type of remote login called secure shell. The computers in the lab (and most other computers on campus) all have a program named "Secure Shell Client." This program can be found under 'Start' 'All Programs' 'General Apps' 'SSH Secure Shell.' If not, you can download the Putty application from the web and execute it instead. After opening this program, you will need to connect it to Wolf. To do this, click "Quick Connect," or just hit the space bar. A new window will ask you for some information. The host name is "wolf0.byu.edu" and the user name for this class is "charmmlab." Once you have entered this information, click connect. Enter the password, charmm, and you should be connected. The screen you now see is a Linux shell that is being run from Wolf.

Directories have been created for each of you named by your first name. Change directories into your directory using the cd command. You can list the contents of your directory using the `ls` command. You should see five files that you will need to complete this lab, including a copy of this document. You will also need to write some new files from scratch. To do this you will need to understand how to use a Linux text editor.

Vi

Like Windows, Linux has many different options for editing text. Unlike Windows, however, these editors do not rely on the mouse. The Linux text editor **vi** hardly uses the mouse at all. To open vi type "vi filename" in the command line of your Linux shell and push enter. There are two modes in vi, command mode and insert mode. The default mode is command mode. In command mode, you use the keyboard to enter commands. For example :w writes the revised version of the file to disk (using the filename you opened with the vi command); :w "filename"is like using "save-as" in Windows: it stores you're file under a new name without disturbing the original. Most of the time you will be using insert mode to enter text. To enter **insert mode** push "i". To exit insert mode, so that you can save your file and quit vi, push the "esc" key. When editing using vi, scroll using the up/down arrows or page-up/page-down on the keyboard.

Here is an example of how to use vi to enter in a new file and save it: first type "vi myfirstfile" to open a new file named myfirstfile. Next enter insert mode, so that you can enter text, by pushing "i." Now type some text and note how the backspace and delete keys work. Once you are done entering text, exit insert mode to return to command mode in order to save your file. Exit insert mode by pushing "Esc." To save your file, push the colon and then w (you don't have to specify a name because you already named

it myfirstfile when you opened it). Once your file is saved, push colon and then q to quit vi. If you want to quit without saving changes, use :q!. If you just want to quit AND save changes, use ZZ.

In the shell command line, list the contents of your directory with the ls command to verify that you created a new file. The long listing (or ls –l) is more helpful than the default listing: gives more info.

II. CHARMM Basics

CHARMM is a command line program. This means that it is runs from a shell window with typed commands. Each command can be typed into the command line one at a time or each command can be read from a script file (commonly referred to as a **stream file** and given the extension ".str"). Like computer languages, CHARMM has a certain vocabulary and syntax to be learned. Some syntax in CHARMM is forgiving. For example, CHARMM commands can be truncated to their first four letters. Also, empty lines and multiple spaces between commands or pieces of commands are know as white space and are ignored by CHARMM.

Other syntax in CHARMM, however, is extremely sensitive. For example, if you tell CHARMM to open a file, the filename will be converted to all lower case unless you put the name in quotes. This can be a problem because Linux is case sensitive. This means that if you want to open the file "MyFile," and you leave off the quotes, CHARMM will look for the file "myfile" which is not the same as "MyFile" in Linux. The CHARMM dictionary and online documentation (www.charmm.org) offer a list of available commands and options. To see how CHARMM commands are used, we'll start off by creating a simple polypeptide, finding its energy, and applying an energy minimization to it. Rather than typing each of these series of commands in one at a time, we will create a stream file that contains all of the options.

CHARMM doesn't even enter into the picture until we are completely finished with the script, at which time we'll invoke CHARMM. So, remember during the next few minutes, as you type in the commands explained in this section, that you're only using the editor program, and that no calculations have taken place until the script is submitted. Using vi, open a new file named "ptrpmin.str"; this will be where your script gets written.

You won't be able to answer the questions in this lab without writing and executing the CHARMM script first.

Let's start by talking you through the initial set of commands summarized at the bottom of the next page.

Writing CHARMM scripts

1. All CHARMM scripts begin with a title. This is useful later in identifying old scripts. Title lines begin with an asterisk. (Also in non-title lines, exclamation points can be used to initiate comments.) The title must finish with a line consisting of only one asterisk, so type:

```
* To create poly-tryptophan and calculate its energy
*
```

CHARMM already knows what atoms and amino acids look like, and how to connect them (i.e., atom masses and charges, bond angles and strengths, etc. are stored in the topology and parameter files). In the next lab, we'll learn how to write an Residue Topology File (RTF) that creates an amino acid from

scratch; for now, let's use the ones the CHARMM programmers wrote. In order for you to make reference to a TRP residue, CHARMM must have first called up all these pre-calculated residue structure and parameter files. The instructions for calling up this stored information have already been written for you in a file called "TOPPARM.STR."

2. Any CHARMM commands can be stored in a file and read into CHARMM by using the STREAM command (miscom.doc). Your first instruction to CHARMM should be:

```
STREAM TOPPARM.STR
```

Filenames enclosed in quotes, such as "TOPPARM.STR," are case-sensitive in CHARMM. This means that if you type "topparm.str," it won't work.

3. Now we're ready to build our 5-mer of tryptophan. The command READ SEQUENCE (struct.doc) tells CHARMM to read a sequence of residue topologies from the RTFs previously loaded. We specify CARD because the sequence is going to be a list of ASCII characters. READ SEQUENCE is always followed by 1) a title (prefaced by asterisks, just like before), 2) the number of residues, and 3) the residue names.

Now we've primed CHARMM for the next command, GENERATE, which constructs one segment in the **Principle Structure File (PSF)** from the residues we specified with READ SEQUENCE. GENERATE must be followed by a name for the segment, which in this case is "PTRP." A segment is any group of residues. Imagine modeling hemoglobin, for example, which has four separate subunits. Each subunit would be a different segment. Once all 4 segments (monomers) are generated, the PSF would contain the whole protein. Water and salt segments might be added to create the proper environment. Organic solvent, lipid, DNA, ligand molecules, or sets of such molecules are other common segments. If we want internal coordinate (IC) tables set up, which we do, the option SETUP must also be specified after GENERATE. These internal coordinate tables are "empty" until CHARMM reads the IC PARAMETER (intcor.doc) command, which fills the IC tables with values from the parameter table which we opened and read in 2.

In order to compute the Cartesian coordinates in the next step, the relation of the structure to the origin (0,0,0), must be specified. The IC SEED command places the first atom specified at the origin, the second on the x axis, and the third in the x-y plane, using bond lengths and bond angles from the IC table.

IC BUILD computes the Cartesian coordinates from the data in the internal coordinate tables (which we just filled, thanks to IC PARAMETER).

Seems like that would be the end of things, except for the cleanup crew in the form of IC PURGE. This command deletes the IC table entries which contain undefined atoms. Remember that each residue includes an amino-terminus and a carboxyl-terminus. Now, connecting these residues with peptide bonds is going to get rid of some hydogens and oxygens, but IC PARAMETER doesn't know that; it gives us bond and energy information for those extra atoms, too. IC PURGE gets rid of this junk.

Now we want to see the result of our troubles so we ask CHARMM to show us what the IC table looks like, with the command PRINT IC. The information in the IC table is presented in columns labeled:

I J K L R(I-J or I-K) T(I-J-K or I-K-J) PHI T(J-K-L) R(K-L)

where I, J, K, and L are atom names, R stands for bond length, T stands for theta (angle), and PHI is the dihedral angle. If an asterisk (*) is present in front of the third atom name, this indicates an improper dihedral in the "PHI" column and R (I-K) and T (I-K-J) are to be used.

The whole set of commands needed to build the PSF for the segment called PTRP and then print the contents of the IC table looks like this:

```
READ SEQUENCE CARD
* Sequence for poly-tryptophan
*
5
TRP TRP TRP TRP TRP
GENERATE PTRP SETUP
IC PARAMETER
IC SEED 1 CA 1 C 2 N
IC BUILD
IC PURGE

PRINT IC
```

(Later, after you execute your CHARMM script, you will come back to this question.)

Question 1: From the contents of the IC table, what is the angle between 1 CG, 1 CD2, and 1 CE2? What is the length of the bond between 4 CD2 and 4 CG? (Clue: is there a bond between those two atoms? If not, why not?)

4. We said at the outset that we were interested in finding the energy of this peptide. Typing ENER (energy.doc) will calculate the potential energy -- you will learn how a bit later in the course and we could leave it at that. But an important parameter in calculating energy is the cutoff distance for non-bonded interactions: in other words, how far apart do two atoms have to be before their effect on each other is deemed 'negligible' and ignored? The following three commands will compute the potential energy three times, each time with different parameters:

```
ENER
ENER CUTNB 5.0 CTONNB 4.0 CTOFNB 4.5
ENER CUTNB 8.0 CTONNB 6.5 CTOFNB 7.5
ENER CUTNB 15.0 CTONNB 11.0 CTOFNB 14.0
```

In the first line we just type ENER. CHARMM would use the default values.

Question 2: From the information that follows the ENER commands in your LOG file (see VI and VII), what is the total energy in each case? What is the largest contributor to the energy? (Ignore the "WARNING" messages for now.)

CHARMM potential energy computation depends on the cutoff parameters. The bigger these parameters the more pairs of atoms are included in the computation. But that does NOT necessarily mean a bigger energy (why?).

5. "**Energy minimization**" and "geometry optimization" are synonymous terms for algorithms that change the coordinates of the peptide (i.e., its shape) to result in a lower potential energy structure. Because we are going to change the coordinates of the peptide, let's keep a copy of the original structure in the **comparison set**, which is a secondary coordinate file where active coordinates can be kept in CHARMM (corman.doc):

```
COORDINATE COPY COMP
```

6. One method of minimization which will be discussed in lectures to come is **Adopted-Basis Newton-Raphson (ABNR)** (minimiz.doc). The following command instructs CHARMM to perform 25 steps of ABNR minimization on our PTRP:

```
MINIMIZE ABNR NSTEP 25 NPRINT 1
```

Question 3: Judging from the lines that follow the MINIMIZE command, what does the command "NPRINT" do?

7. CHARMM has temporarily stored the original coordinates of poly-TRP (in the COMParison coordinate set) and the coordinates of the new minimized structure (in the MAIN coordinate set). If we had CHARMM run this script and then finish, both sets of coordinates would be lost, unless we save them to permanent, humanly readable files on disk. To do this we have to start by designating a **FORTRAN logical unit number** (miscom.doc). This unit number could be any number from 1 – 99, but don't use 5 or 6. These are standard input (the keyboard) and standard output (the monitor). Don't use the same unit number twice in a script unless the first unit is closed or you want to write again to the same file. We also specify that the file type will be ASCII, with the keyword CARD. Then we name the file, adding the suffix ".CRD" as we do with any set of coordinates. We haven't actually saved the coordinates to this file yet; we've just named it. To save coordinates to any unit, the command is WRITE COOR, or in the case of coordinates stored in the COMP set, WRITE COOR COMP. Then, just to be neat, we close the unit. If you forget this on occasion, CHARMM's sub-program RDTITL will do it for you.

```
OPEN WRITE UNIT 10 CARD NAME "ptrp.crd"
WRITE COOR COMP UNIT 10 CARD
CLOSE UNIT 10

OPEN WRITE UNIT 20 CARD NAME "ptrpmin.crd"
WRITE COOR UNIT 20 CARD
CLOSE UNIT 20
```

How did poly-tryptophan change upon ABNR minimization? To find out we should compare the two coordinate sets, both in Cartesian coordinates and in internal coordinates.

8. The command to calculate the difference between the internal coordinates (preminimization minus postminimization) is:

```
IC DIFF
```

The command to calculation the difference between Cartesian coordinates (preminimization minus postminimization) is:

```
COOR DIFF
```

9. Now, to print out these differences to the log file, we print out the MAIN Cartesian coordinate set and the IC table, just like we did above to print the coordinates themselves. The original coordinates in these two tables have all been replaced by differences in the coordinates!

```
PRINT COOR
PRINT IC
```

Question 4: Judging from the IC table here and above (before minimization), what changed in the structure?

Question 5: The structure seems to be shrinking, but not much. (To see this, check the change in distance between two atoms on opposite ends of the peptide). Why? (Hint: think about the absence of solvent or other molecules in the environment.)

10. Finally, be sure to tell CHARMM that your stream file is finished:

```
STOP
```

11. Now save your file and exit vi. We will next run the script in CHARMM.

Running scripts

As I mentioned in the first section, CHARMM often uses more than one processor. Wolf consists of 8 processors that are each split into two virtual processors for a total of 16 processors. CHARMM can use any, part, or all of these 16 processors when specified. To help us use these processors efficiently, wolf has a scheduler program named **SLURM** that distributes processes evenly over the processors so that many jobs can be run at once without interfering with each other. While SLURM is simple to use, a shell script has been written to simplify the CHARMM script submission process. The shell script is named submit.sh and a unique copy of it is in each of your directories.

To submit a CHARMM script, open the submit.sh file and change the value of "STREAMFILE" from test.str to the name of your script, as well as the value of "DIR" to your working directory. Save the modified file and return to the shell command line. Now all you have to do to submit your script is type `./submit.sh` and push enter. This will submit your job. When CHARMM is done, you will notice that there are several new files in your directory. These new files will be the ones you told CHARMM to write and two new ones named slurm-??? and yourstreamfile.str.log. The SLURM file is error messages from slurm and the log file is from CHARMM.

In our stream file, we asked CHARMM to give us some information such as coordinates and energies. These values are recorded in a **log file**. Log files contain not only the information that you requested, but also all the interaction between different subprograms in CHARMM. You'll recognize your own ptrpmin.str commands showing up as part of these conversations.

With the log file open, go back and answer questions 1-5, then finish the remaining questions.

Question 6: What are the components of the energy? Does the system have any kinetic energy?

Question 7: What changed in coordinates and potential energies upon minimization of PTRP?

CRD files: Open "ptrp.crd" and "ptrpmin.crd." Again, if they don't open, you might try holding 'Ctrl' while clicking on the file name. Review the structure of the coordinate files, noting how they compare to the output in the logfile from PRINT COOR and notice how the x, y, and z coordinates changed for a few of the atoms.

To finish, send an email to your TA containing the answers to the questions in the lab.

LAB 3: MOLECULAR STRUCTURE MANIPULATION

The Introduction to CHARMM lab previewed how to create molecules from scratch, using READ SEQUENCE, GENERATE, and IC PARAMETERS and IC BUILD in CHARMM. Unfortunately, it's not often that a structure made straight from an RTF is in a biologically relevant form. This lab, as well as others, will teach you how to make a structure ready for simulations and visualization.

I. Basic Commands

As was mentioned in Introduction to CHARMM lab, the language of CHARMM consists of many one- or two-word commands. You already know OPEN, CLOSE, WRITE COOR, and the commands needed to build a PSF. In this lab, we take you through a few more of the absolute basics. After you have learned these commands, you can start writing programs in CHARMM. If you ever have any questions about these or any other commands, your first stop should be the CHARMM Command Dictionary or www.charmm.org.

Wildcards, Selections, and Variables

In both UNIX and CHARMM, the asterisk "*" can sometimes be used as a **wildcard** to mean "any set of characters." See the definition of SELECT...END below for an example of the wild card in CHARMM.

Imagine having a structure consisting of several segments, but wanting to apply the ENERGY command to only one, or wanting to know the COOR DIFF between only certain atoms in a structure. The set of atoms to which most commands are applied can be shrunk with an atom **selection** using the SELECT command (select.doc). You can specify the subset by using the keywords ATOMS, SEGID (segment name), ISEG (segment number), RESN (residue name), or another keyword (see the CHARMM Dictionary, Volume II, pg. 100 for the whole list). Most often, we use the SELECT ATOM command, which is always followed by 1) the segment ID, 2) the residue ID, 3) the atom name (though the documentation says type), and 4) END. Selections are passed as arguments to various commands within CHARMM. A selection argument of all atoms of a segment called SEG1 for example, would be as follows:

```
SELECT ATOM SEG1 * * END
```

*When two selections are called for, the entire SELECT...END command must be repeated twice. To select all atoms of the first residue of SEG1 for the first selection, and second residue of SEG1 for the second selection, the command would be SELECT ATOM SEG1 1 * END SELECT ATOM SEG1 2 * END.*

There will be occasions when you will need to use a **variable** in a CHARMM script. The SET command assigns a value to a variable. For instance, "SET dist 15.3" gives the variable named "dist" a value of 15.3. When you wish to use the value of a variable where CHARMM is expecting a number, you use the variable with a "@" sign in front of it. CHARMM then reads the command containing the variable, as though it contained the value of that variable. An example of this is demonstrated in the following sequence of commands:

```
SET unlucky 13
OPEN READ UNIT 10 CARD NAME "SPIFFY@unlucky.CRD"
```

The second command will be interpreted by CHARMM as:

```
OPEN READ UNIT 10 CARD NAME "SPIFFY13.CRD"
```

Variables can also be SET to the values of other internal CHARMM variables. For example, **SET TEMP ?ENER** would set the variable "TEMP" to the total potential energy computed most recently. The command **SET TEMP MINIMIZED** would set the variable "TEMP" to contain the word (or string, for you programmers) "MINIMIZED."

You can manipulate the values stored in variables using INCR and DECR. The syntax for both of them is the command (INCR or DECR), the variable name, and the amount to add or subtract. The following command will add 10 to the value stored in number so that the final value of "number" is 10:

```
SET number 0
INCR number 10
```

Charmm also supports referencing variables in an array-like fashion. For example, if you set a variable with a constant prefix and integer suffix, you can access various variable values later using an "@" in front of the variable name prefix followed by "@@" and the array index you wish to access. Here is an example:

```
SET segName1 M2A
SET segName2 M2B
SET segName3 M2C
SET segName4 M2D
SET index 3
SET newVariable @segName@@index
```

In the above example, four sequential variables with the prefix "segName" are set to four different strings. To access the third string in the sequence, or array, of variables, a variable "index" is set to "3" and then "newVariable" is set to the value of the third variable in the array, "M2C" by passing "@segName@@index."

Conditional statements

Often **conditional statements** and loops are handy in CHARMM scripts (miscom.doc). If you have some experience with FORTRAN or the C programming language, the ideas are the same.

Sometimes you only want to run a command if a certain condition is true. Let's say you minimize your structure for 200 steps of ABNR, and then you want to minimize it again if the total energy is greater than a given value (say, -100 kcal/mol). In this case you could use an if statement. Here's how you could do it:

```
MINIMIZE ABNR NSTEP 200
SET MYENERGY ?ENER
IF MYENERGY GT -100 MINIMIZE ABNR NSTEP 200
```

The condition in this example was the greater than (GT) conditional statement. If the value stored in MYENERGY is greater than -100, then CHARMM will execute 200 more steps of minimization. If the value stored in MYENERGY is not greater than -100, then CHARMM will skip over the additional minimization. Besides "greater than", GT, we could use the conditionals EQ ("equal to"), NE ("not equal to"), GE ("greater than or equal to"), LT ("less than"), or LE ("less than or equal to"). "IF" is always followed by a variable (not preceded by a "@"), a conditional, a reference (either a value or another variable proceeded by a "@"), and a CHARMM command.

Loops

A **loop** is a piece of a program designed to repeat itself many times (miscom.doc). Loops can be created using the GOTO and LABEL commands. LABEL designates the beginning of a loop, and is always followed by the name of the loop. GOTO is the command that allows for the repetition (iteration) of the loop. GOTO must also be followed by the name of the labeled line that CHARMM is to be directed to. If we wanted to minimize a structure again and again until the potential energy was less than -100, a loop could be used. Once the energy went below -100, the conditional would not be met, and CHARMM would ignore the GOTO command and go on to the next line. The code would look like this:

```
LABEL LOOP1
MINIMIZE ABNR NSTEP 200
SET MYENERGY ?ENER
IF MYENERGY GT -100 GOTO LOOP1
```

What if we wanted to minimize our structure 200, then 300, then 400 times? We could attack the problem by using 3 different parameters:

```
SET 1 200
SET 2 300
SET 3 400
MINIMIZE ABNR NSTEP @1
MINIMIZE ABNR NSTEP @2
MINIMIZE ABNR NSTEP @3
```

But if we want to increase the number of steps by 100 until the number reached 10000, it would take a long time to enter in all of the commands. Instead of typing 100 commands, you could use a loop.

```
SET n 200
LABEL min_loop
MINIMIZE ABNR NSTEP @n
INCR n BY 100
IF n LE 10000 GOTO min_loop
```

Question 1: What would happen if the SET command were inside the loop?

IC Edit

As you might guess, IC EDIT (intcor.doc) is used to edit internal coordinates. The command can be followed by as many lines as desired: each containing 1) the keywords DIST, ANGLE, or DIHEDRAL, 2) an atom selection of two, three, or four atoms for DIST, ANGL, or DIHE, respectively, taking the form residuenumber atomname, 3) a value in angstroms or degrees for the distance, angle, or dihedral. On another line must appear END. Improper dihedrals, as in the IC table, are indicated by an asterisk

before the third atom specification. To change the length of a bond (DISTANCE) to 2.0 angstroms, for example, IC EDIT is used in the following manner:

```
IC EDIT
DIST 3 N 3 C 2.0
END
```

Write Title

Sometimes CHARMM won't let you write to a file unless it has a title first. Other times you really need the title, to include information about the data in the file. WRITE TITLE (miscom.doc) is always followed by the information you want at the top of the file, preceded by asterisks:

```
WRITE TITLE
* This is where you describe the file
*
```

WRITE TITLE really gets useful when you want to report the values of internal CHARMM variables not normally reported in the format you want. Imagine doing several energy calculations on different structures, and wanting a concise, sequentially numbered list of the energies. Inside a loop, you might write:

```
WRITE TITLE
* @num ?ENER
*
```

This will write the value of CHARMM variable "num" followed by the potential energy of the current structure to standard output, which is the monitor--unless it was redirected to an output file in the CHARMM command line, (or unless you have opened a file on a unit for writing, as exemplified in line 22 of ALPHAHLX.STR below, and you have specified the unit number in the WRITE TITLE line; but don't bother with this approach for now).

II. Sample Scripts

Question 2: Look at the two scripts on the following pages (ALPHA.STR and ALPHAHLX.STR). The numbers at the far left are for convenience and should not appear in the CHARMM-ready script. Wherever the number at the far left is bold and underlined, tell what that command is doing and, where applicable, what sort of information is contained in the file it makes reference to. (Lab 2 may be helpful here.)

Lines 1-6 are the content of "TOPPARM.STR," the function of which was discussed in Lab 2. Make note of them since you'll need them in Question 3.

```
************************ALPHA.STR************************
* Copyright 1988 Polygen Corporation
*This input file constructs an alpha helix for polyalanine
*

1 OPEN READ UNIT 11 CARD NAME ~/top_all27_prot_lipid.rtf
2 READ RTF UNIT 11 CARD
3 CLOSE UNIT 11

4 OPEN READ UNIT 11 CARD NAME ~/par_all27_prot_lipid.prm
5 READ PARA UNIT 11 CARD
6 CLOSE UNIT 11

7 READ SEQUENCE CARD
8 * Polyalanine
9 *
10      11
11      ALA ALA ALA ALA ALA ALA ALA ALA ALA ALA ALA
12      GENERATE HELX SETUP
13      IC PARAMETERS

14      SET FSTRES 2     ! First residue to be modified.
15      SET LSTRES 10    ! Last residue to be modified.

16      SET PHI -57.0
17      SET PSI -47.0

18      OPEN READ UNIT 18 CARD NAME "ALPHAHLX.STR"
19      STREAM UNIT 18

20      IC SEED 1 N 1 CA 1 C
21      IC BUILD
22      OPEN WRITE UNIT 40 CARD NAME "ALPHAHLX.PDB"
23      WRITE COOR PDB CARD UNIT 40

24      STOP
```

```
**********************APHAHLX.STR*****************************
* Copyright 1988 Polygen Corporation
* This stream file edits the internal coordinate table by defining
* the phi and psi angles to be an alpha helix for a range of residues
(CHARMM
* variables FSTRES to LSTRES)
*
! Define variables for residues before and after current residue
1 SET RES @FSTRES
2 CALC NXTRES = @RES+1
3 CALC PRERES = @RES-1


4 LABEL START

! Invoke IC EDIT mode, and define phi and psi dihedral angles
! for an alpha helix

5 IC EDIT
6 DIHE @PRERES C @RES N @RES CA @RES C @PHI
7 DIHE @RES N @RES CA @RES C @NXTRES N @PSI

8 END

! Increment counters and check for last specified residue

9 INCR PRERES BY 1
10      INCR RES BY 1
11      INCR NXTRES BY 1
12      IF RES LE @LSTRES GOTO START
```

With the previous scripts as a model, answer the following question.

Question 3: After the first iteration of the loop in ALPHAHLX.STR (just after line 11), what are the values of FSTRES, LSTRES, RES, NXTRES, and PRERES? How would CHARMM interpret lines 6 and 7 of "ALPHAHLX.STR (what would the command look like after CHARMM substituted values for the parameters)?"

Question 4: Why is ALPHA.STR set up to modify the dihedrals of residues 2 through 10 and not 1 or 11? (Hint: What would happen in ALPHAHLX.STR if res1 had been set to 1 or res2 had been set to 12?) .

Question 5: How would you change this script to build a 3-10 helix with 14 ALA residues, given that in this sort of helix, φ = -49 and ψ = -26? How is it different from an α-helix?

Copy the files for this lab into your personal directory. ALPHA.STR and ALPHAHLX.STR have been provided for you, with the line numbers omitted. Run ALPHA.STR with CHARMM on Wolf0 to create alphahlx.pdb. Import this PDB into VMD and examine it. Create the 3-10 helix PDB file (change

ALPHA.STR to make "3_10.STR" which in turn creates "3_10.PDB"). Import this "3_10.PDB" into VMD. Examine the difference between 3-10 and α-helices and describe them in your answer to Question 5.

III. Debugging

1. If the writer of a stream file makes a mistake somewhere, CHARMM will often stop before finishing the task. If this happens, there will be a message about 15 lines from the end of the ".LOG" file saying, "Execution terminated due to the detection of a **fatal error**." The process of reading the ".LOG" file and trying to decipher what the problem is called debugging, and it consumes most of the lab time of every computer scientist in existence and most molecular modelers too. To witness this gruesome spectacle, read "ALPHA_MESSED.STR." You will find this file after you run ALPHA.STR. It will be with all of the other job files. This is the same program as the one you commented in Question 2, but if you look carefully at line 14 you'll see a mistake ("SETR" should be "SET"). Save "ALPHA_MESSED.STR" to your computer, rename it "MESSED.STR," and run it.

When "MESSED.STR" is finished, read the log file. The last lines, below the one starting "MAXIMUM SPACE USED IS…," are normal information lines given in every log file whether CHARMM has abnormally terminated or not. The "fatal error" message just above them is what we're interested in: "Unrecognized command: SETR." If we were writing a stream file, this message would tell us that something was wrong with the SETR command in line 14. We'd have to go back to the stream file, change that command to SET, and run our script again. Of course, if our change was also incorrect, then CHARMM would stop this time as well. When debugging, it is often necessary to repeat this process many times in order to execute the file correctly. The most common bugs are often simple typos. For this reason, it is a good idea to take your time when typing scripts.

2. In this context of bugs and fatal errors, it's important to take a minute to talk about BOMLEV. We just saw CHARMM die when it hit a mistake in "MESSED.STR." CHARMM doesn't die every time it reads something wrong. In the Introduction to CHARMM lab, for example, CHARMM didn't think that our choice of non-bonded cutoffs was appropriate, so it gave us a "WARNING" (check "TRPMIN.LOG" from last week if you don't remember this). In fact, every error in CHARMM programming has a BOMLEV (miscom.doc) associated with it, ranging from 5 to -5. The negative levels are pretty severe and the positive, not so bad. In "MESSED.STR" you can see that an unrecognized command is level 0. The command BOMLEV dictates at what level error CHARMM crashes. Right now BOMLEV is set at the default, zero.

Managing your BOMLEV is sort of a tricky business, and you can be content to use 0 for now. To show you a bit of the complexity involved, first, correct line 14 of "ALPHA_MESSED.STR." Then, change line 16 from "ALPHAHLX.STR" to "ALPHAHLX." Now, there's no such file as "ALPHAHLX" in your directory, so when you tell CHARMM to read this stream file, it won't be able to read anything. Save your file and run "ALPHA_MESSED.STR" again. Look carefully at the new log file.

Although CHARMM was not able to read "ALPHAHLX," the program doesn't crash because the BOMLEV wasn't low enough. Of course, without the stream file, "MESSED.STR" won't change the dihedrals of our polypeptide to build an alpha-helix. This is why you must always read your log files even if CHARMM doesn't crash openly. This raises a question: if it's such a nuisance to allow CHARMM to continue even after an error like this, why not raise the BOMLEV? If we did raise the BOMLEV,

CHARMM would have crashed in the Introduction to CHARMM lab when it disliked our non-bonded cutoffs, and that would have been a nuisance too.

IV. Good Programming Practice

One indicator of good computer programmers is the readability of their code. This applies to molecular modelers as well. There are several small tricks that help humans be able to understand scripts.

White Space

If this lab had been written as one long paragraph without tabs or blank lines, it would have been harder to read. Scripts can also be easier to read if they contain an appropriate usage of white space. Extra tabs, spaces, and line breaks are considered white space. In stream files, CHARMM does not care if you have one blank space or twenty blank spaces, one blank line or 4 blank lines, they are run the same. Format your scripts so that they are easy to read.

Comments

The "!" character is the comment character in CHARMM stream files. This means that anything that comes after "!" is not read by CHARMM. Take a minute and look at "top_all27_prot_lipid.rtf." Compare the residues defined in this file to those printed at the end of this lab. You will notice that in the "top_all27_prot_lipid.rtf" file, there are diagrams of each residue. CHARMM does not read these diagrams because of the "!" on each line. Comments are a great way to explain a cryptic part of your script to make it more readable.

Variable Names

When you name variables in your script, make them short and descriptive. For example, if you need a variable to count the number of molecules outside a particular region, you could call it NumInBox or num_in_box. Notice the capitalization. CHARMM is not case-sensitive, meaning it does not distinguish between NumInBox and numinbox. Variations in capitalization can be used to make the program more readable. In these labs, we are using the old-fashioned tradition of capitalizing everything as a syntactical convention to make it easier for you to read the handout. Notice the difference shown between examples 1 and 2 below.

```
! Example 1
SET 1 6
SET 2 10
SET 3 4
SET 4 5
SET 5 1
LABEL THINGY
INCR 2 BY @2
INCR 3 BY @3
INCR 4 BY @4
INCR 5 BY 1
IF 5 LE @1 GOTO THINGY
INCR 1 by @2
INCR 1 by @3
```

```
! Example 2
! script computes x(a+b+c)
SET x 6
SET a 10
SET b 4
SET c 5

! Add a, b, and c x times
SET count 1
LABEL start
INCR a BY @a
INCR b BY @b
INCR c BY @c
INCR count BY 1
IF count LE @x GOTO start

! sum the products
SET answer @a
INCR answer by @b
INCR answer by @c
```

Question 6: Write a script that builds a polypeptide ALA-GLU-TRP and produces three "PDB" files called "ALA.PDB," " GLU.PDB," and "TRP.PDB" which contain only the atoms in the first, second, or third residues, respectively. Use the SELE, ATOM, SET, and LABEL (loop) commands in your script. Use a separate stream file and call it "ala_glu_trp.str".

When writing this script, keep the sheet from the Introduction to CHARMM lab close by your side. Don't forget, right after your title, your first commands to CHARMM must be lines 1-6 from "ALPHA.STR," which are the commands in "TOPPARM.STR." One more reminder: you must close a unit if you wish to use its number again.

Run "ala_glu_trp.str." Read "ala_glu_trp.log". If you have any errors, make an effort to correct the problem on your own. Go back to your "ala_glu_trp.log" and try to see what went wrong (look for typos, and make sure you phrased things exactly as they're listed in previous labs), make a change, and rerun. Comment the file appropriately. When you submit your answers for this lab to the T.A., please attach your copy of the working stream file.

V. RTF files

A **Residue Topology File (RTF)** (rtop.doc) is a collection of information about the charge, connectivity, and arrangement in space of a polymeric residue or a molecule. On the attached pages are some sample RTFs for different amino acids. The GROU lines are used for simplifying electrostatic calculations when needed.

The first lines, which begin with ATOM, are quite simply a list of all the atoms that the residues (and molecules) in the file could contain. The atom list information is in the order ATOM atomname atomtype partial charge.

AUTOGENERATE ANGLES causes the angle list to be generated from the bond list.

The lines which begin with BOND determine which atoms are covalently bonded. The bond information is in the form BOND atomname (i) atomname (j). Don't get confused if there are several of these in a row. The purpose of these lines is to create listed of bonded interactions for the ENERGY function (not to specify the internal coordinates – that is the purpose of the IC lines alone, which are found below).

Lines that begin with DOUBLE specify double bonds.

There are no ANGLE lines because those are AUTOGENERATED from the BOND list.

The lines that begin DIHE specify all the dihedral angles that will contribute to the ENERGY.

The lines that begin IMPR specify "improper dihedral" relationships, which are artificial energy terms (or "penalties" if you like) used to enforce miscellaneous features of the molecular geometry related to electron structure such as chirality or planarity of conjugated ring structures.

Following these lines are lists of hydrogen bond donors and acceptors, which are only used in the hydrogen-bond detection facility, and not for energy calculations.

The remainder of the residue specification is a list of IC lines. Recall from the Introduction to CHARMM lab that the numbers represent bond length, angle, dihedral, angle, and bond length. Many times, all but the dihedral angle are given as 0.0 in the RTF. Here the numbers serve as placeholders. Any 0.0 will be replaced by the parameter file value for the corresponding atom types when IC PARAM is executed prior to IC SEED and IC BUILD.

> *Question 7: Identify residues A through E.* Use your intuition, focusing first on the atoms that are listed as bonded together and the atom types in the ATOM lines. You may want to open top_all27_prot_lipid.rtf to check the atom types and their typical usage at the top of the file.

Now that you have a bit of experience in deciphering RTFs, it's time to try something a little tougher: writing one. This will give you a better idea of what is contained in these files and will provide valuable practice.

> *Question 8: Given the RTFs for phosphotyrosine and threonine, create an RTF for phospho-threonine. (Hint: threonine should be phosphorylated in a way analogous to tyrosine.) You are not expected to fill in the IC tables portion of the RTF, though you are welcome to attempt it if you desire.*

When you are done answering all of the questions, email your answers to your T.A. Be sure to attach "ala_glu_trp.str."

```
RESI ??A??         0.00
GROUP
ATOM N    NH1    -0.47
ATOM HN   H       0.31
ATOM CA   CT1     0.07
ATOM HA   HB      0.09
GROUP
ATOM CB   CT3    -0.27
ATOM HB1  HA      0.09
ATOM HB2  HA      0.09
ATOM HB3  HA      0.09
GROUP
ATOM C    C       0.51
ATOM O    O      -0.51
BOND CB CA  N  HN  N  CA
BOND C  CA  C  +N  CA HA  CB HB1  CB HB2  CB HB3
DOUBLE O  C
IMPR N -C CA HN  C CA +N O
DONOR HN N
ACCEPTOR O C
IC -C   CA   *N   HN    1.3551 126.4900  180.0000 115.4200  0.9996
IC -C   N    CA   C     1.3551 126.4900  180.0000 114.4400  1.5390
IC N    CA   C    +N    1.4592 114.4400  180.0000 116.8400  1.3558
IC +N   CA   *C   O     1.3558 116.8400  180.0000 122.5200  1.2297
IC CA   C    +N   +CA   1.5390 116.8400  180.0000 126.7700  1.4613
IC N    C    *CA  CB    1.4592 114.4400  123.2300 111.0900  1.5461
IC N    C    *CA  HA    1.4592 114.4400 -120.4500 106.3900  1.0840
```

```
IC C    CA   CB   HB1   1.5390 111.0900  177.2500 109.6000  1.1109
IC HB1  CA   *CB  HB2   1.1109 109.6000  119.1300 111.0500  1.1119
IC HB1  CA   *CB  HB3   1.1109 109.6000 -119.5800 111.6100  1.1114

RESI ??B??         0.00
GROUP
ATOM N    NH1    -0.47
ATOM HN   H       0.31
ATOM CA   CT1     0.07
ATOM HA   HB      0.09
GROUP
ATOM CB   CT2    -0.18
ATOM HB1  HA      0.09
ATOM HB2  HA      0.09
GROUP
ATOM CG   CY     -0.03
ATOM CD1  CA      0.035
ATOM HD1  HP      0.115
ATOM NE1  NY     -0.61
ATOM HE1  H       0.38
ATOM CE2  CPT     0.13
ATOM CD2  CPT    -0.02
GROUP
ATOM CE3  CA     -0.115
ATOM HE3  HP      0.115
GROUP
ATOM CZ3  CA     -0.115
ATOM HZ3  HP      0.115
GROUP
ATOM CZ2  CA     -0.115
ATOM HZ2  HP      0.115
GROUP
ATOM CH2  CA     -0.115
ATOM HH2  HP      0.115
GROUP
ATOM C    C       0.51
ATOM O    O      -0.51
BOND CB  CA   CG  CB   CD2 CG   NE1 CD1
BOND CZ2 CE2
BOND N   HN   N   CA     C   CA   C   +N
BOND CZ3 CH2  CD2 CE3  NE1 CE2  CA  HA   CB  HB1
BOND CB  HB2  CD1 HD1  NE1 HE1  CE3 HE3  CZ2 HZ2
BOND CZ3 HZ3  CH2 HH2
DOUBLE  O   C   CD1 CG   CE2 CD2  CZ3 CE3  CH2 CZ2
IMPR N -C CA HN  C CA +N O
DONOR HN N
DONOR HE1 NE1
ACCEPTOR O C
IC -C   CA   *N   HN    1.3482 123.5100  180.0000 115.0200  0.9972
IC -C   N    CA   C     1.3482 123.5100  180.0000 107.6900  1.5202
IC N    CA   C    +N    1.4507 107.6900  180.0000 117.5700  1.3505
IC +N   CA   *C   O     1.3505 117.5700  180.0000 121.0800  1.2304
```

```
IC CA   C    +N   +CA   1.5202 117.5700  180.0000 124.8800  1.4526
IC N    C    *CA  CB    1.4507 107.6900  122.6800 111.2300  1.5560
IC N    C    *CA  HA    1.4507 107.6900 -117.0200 106.9200  1.0835
IC N    CA   CB   CG    1.4507 111.6800  180.0000 115.1400  1.5233
IC CG   CA   *CB  HB1   1.5233 115.1400  119.1700 107.8400  1.1127
IC CG   CA   *CB  HB2   1.5233 115.1400 -124.7300 109.8700  1.1118
IC CA   CB   CG   CD2   1.5560 115.1400   90.0000 123.9500  1.4407
IC CD2  CB   *CG  CD1   1.4407 123.9500 -172.8100 129.1800  1.3679
IC CD1  CG   CD2  CE2   1.3679 106.5700   -0.0800 106.6500  1.4126
IC CG   CD2  CE2  NE1   1.4407 106.6500    0.1400 107.8700  1.3746
IC CE2  CG   *CD2 CE3   1.4126 106.6500  179.2100 132.5400  1.4011
IC CE2  CD2  CE3  CZ3   1.4126 120.8000   -0.2000 118.1600  1.4017
IC CD2  CE3  CZ3  CH2   1.4011 118.1600    0.1000 120.9700  1.4019
IC CE3  CZ3  CH2  CZ2   1.4017 120.9700    0.0100 120.8700  1.4030
IC CZ3  CD2  *CE3 HE3   1.4017 118.1600 -179.6200 121.8400  1.0815
IC CH2  CE3  *CZ3 HZ3   1.4019 120.9700 -179.8200 119.4500  1.0811
IC CZ2  CZ3  *CH2 HH2   1.4030 120.8700 -179.9200 119.5700  1.0811
IC CE2  CH2  *CZ2 HZ2   1.3939 118.4200  179.8700 120.0800  1.0790
IC CD1  CE2  *NE1 HE1   1.3752 108.8100  177.7800 124.6800  0.9767
IC CG   NE1  *CD1 HD1   1.3679 110.1000  178.1000 125.4300  1.0820

RESI ??C??         0.00
GROUP
ATOM N    NH1    -0.47
ATOM HN   H       0.31
ATOM CA   CT1     0.07
ATOM HA   HB      0.09
GROUP
ATOM CB   CT1    -0.09
ATOM HB   HA      0.09
GROUP
ATOM CG2  CT3    -0.27
ATOM HG21 HA      0.09
ATOM HG22 HA      0.09
ATOM HG23 HA      0.09
GROUP
ATOM CG1  CT2    -0.18
ATOM HG11 HA      0.09
ATOM HG12 HA      0.09
GROUP
ATOM CD   CT3    -0.27
ATOM HD1  HA      0.09
ATOM HD2  HA      0.09
ATOM HD3  HA      0.09
GROUP
ATOM C    C       0.51
ATOM O    O      -0.51
BOND CB  CA   CG1 CB   CG2 CB   CD  CG1
BOND N   HN   N   CA    C   CA   C   +N
BOND CA  HA   CB  HB   CG1 HG11 CG1 HG12 CG2 HG21
BOND CG2 HG22 CG2 HG23 CD  HD1  CD  HD2  CD  HD3
DOUBLE  O   C
```

```
IMPR N -C CA HN  C CA +N O
DONOR HN N
ACCEPTOR O C
IC -C   CA   *N   HN    1.3470 124.1600  180.0000 114.1900  0.9978
IC -C   N    CA   C     1.3470 124.1600  180.0000 106.3500  1.5190
IC N    CA   C    +N    1.4542 106.3500  180.0000 117.9700  1.3465
IC +N   CA   *C   O     1.3465 117.9700  180.0000 120.5900  1.2300
IC CA   C    +N   +CA   1.5190 117.9700  180.0000 124.2100  1.4467
IC N    C    *CA  CB    1.4542 106.3500  124.2200 112.9300  1.5681
IC N    C    *CA  HA    1.4542 106.3500 -115.6300 106.8100  1.0826
IC N    CA   CB   CG1   1.4542 112.7900  180.0000 113.6300  1.5498
IC CG1  CA   *CB  HB    1.5498 113.6300  114.5500 104.4800  1.1195
IC CG1  CA   *CB  CG2   1.5498 113.6300 -130.0400 113.9300  1.5452
IC CA   CB   CG2  HG21  1.5681 113.9300 -171.3000 110.6100  1.1100
IC HG21 CB   *CG2 HG22  1.1100 110.6100  119.3500 110.9000  1.1102
IC HG21 CB   *CG2 HG23  1.1100 110.6100 -120.0900 110.9700  1.1105
IC CA   CB   CG1  CD    1.5681 113.6300  180.0000 114.0900  1.5381
IC CD   CB   *CG1 HG11  1.5381 114.0900  122.3600 109.7800  1.1130
IC CD   CB   *CG1 HG12  1.5381 114.0900 -120.5900 108.8900  1.1141
IC CB   CG1  CD   HD1   1.5498 114.0900 -176.7800 110.3100  1.1115
IC HD1  CG1  *CD  HD2   1.1115 110.3100  119.7500 110.6500  1.1113
IC HD1  CG1  *CD  HD3   1.1115 110.3100 -119.7000 111.0200  1.1103

RESI ??D??         0.00
GROUP
ATOM N    NH1    -0.47
ATOM HN   H       0.31
ATOM CA   CT1     0.07
ATOM HA   HB      0.09
GROUP
ATOM CB   CT2    -0.18
ATOM HB1  HA      0.09
ATOM HB2  HA      0.09
GROUP
ATOM CG   CT2    -0.18
ATOM HG1  HA      0.09
ATOM HG2  HA      0.09
GROUP
ATOM CD   CC      0.55
ATOM OE1  O      -0.55
GROUP
ATOM NE2  NH2    -0.62
ATOM HE21 H       0.32
ATOM HE22 H       0.30
GROUP
ATOM C    C       0.51
ATOM O    O      -0.51
BOND CB CA  CG  CB   CD  CG   NE2 CD
BOND N  HN  N   CA   C   CA
BOND C  +N  CA  HA   CB  HB1  CB  HB2  CG HG1
BOND CG HG2 NE2 HE21 NE2 HE22
DOUBLE O  C    CD  OE1
```

```
IMPR N   -C  CA   HN    C   CA +N   O
IMPR CD  NE2 CG   OE1   CD  CG NE2  OE1
IMPR NE2 CD  HE21 HE22  NE2 CD HE22 HE21
DONOR HN N
DONOR HE21 NE2
DONOR HE22 NE2
ACCEPTOR OE1 CD
ACCEPTOR O C
IC -C   CA   *N   HN    1.3477 123.9300  180.0000 114.4500  0.9984
IC -C   N    CA   C     1.3477 123.9300  180.0000 106.5700  1.5180
IC N    CA   C    +N    1.4506 106.5700  180.0000 117.7200  1.3463
IC +N   CA   *C   O     1.3463 117.7200  180.0000 120.5900  1.2291
IC CA   C    +N   +CA   1.5180 117.7200  180.0000 124.3500  1.4461
IC N    C    *CA  CB    1.4506 106.5700  121.9100 111.6800  1.5538
IC N    C    *CA  HA    1.4506 106.5700 -116.8200 107.5300  1.0832
IC N    CA   CB   CG    1.4506 111.4400  180.0000 115.5200  1.5534
IC CG   CA   *CB  HB1   1.5534 115.5200  120.9300 106.8000  1.1147
IC CG   CA   *CB  HB2   1.5534 115.5200 -124.5800 109.3400  1.1140
IC CA   CB   CG   CD    1.5538 115.5200  180.0000 112.5000  1.5320
IC CD   CB   *CG  HG1   1.5320 112.5000  118.6900 110.4100  1.1112
IC CD   CB   *CG  HG2   1.5320 112.5000 -121.9100 110.7400  1.1094
IC CB   CG   CD   OE1   1.5534 112.5000  180.0000 121.5200  1.2294
IC OE1  CG   *CD  NE2   1.2294 121.5200  179.5700 116.8400  1.3530
IC CG   CD   NE2  HE21  1.5320 116.8400 -179.7200 116.8600  0.9959
IC HE21 CD   *NE2 HE22  0.9959 116.8600 -178.9100 119.8300  0.9943

RESI ??E??          1.00
GROUP
ATOM N    NH1    -0.47
ATOM HN   H       0.31
ATOM CA   CT1     0.07
ATOM HA   HB      0.09
GROUP
ATOM CB   CT2    -0.18
ATOM HB1  HA      0.09
ATOM HB2  HA      0.09
GROUP
ATOM CG   CT2    -0.18
ATOM HG1  HA      0.09
ATOM HG2  HA      0.09
GROUP
ATOM CD   CT2     0.20
ATOM HD1  HA      0.09
ATOM HD2  HA      0.09
ATOM NE   NC2    -0.70
ATOM HE   HC      0.44
ATOM CZ   C       0.64
ATOM NH1  NC2    -0.80
ATOM HH11 HC      0.46
ATOM HH12 HC      0.46
ATOM NH2  NC2    -0.80
ATOM HH21 HC      0.46
```

```
ATOM HH22 HC      0.46
GROUP
ATOM C    C       0.51
ATOM O    O      -0.51
BOND CB  CA  CG  CB  CD CG  NE CD  CZ NE
BOND NH2 CZ  N  HN  N  CA
BOND C   CA  C  +N  CA HA  CB HB1
BOND CB  HB2 CG  HG1 CG HG2 CD HD1 CD HD2
BOND NE  HE  NH1 HH11  NH1 HH12  NH2 HH21  NH2 HH22
DOUBLE O  C    CZ  NH1
IMPR N  -C  CA  HN   C CA +N O
IMPR CZ NH1 NH2 NE
DONOR HN N
DONOR HE NE
DONOR HH11 NH1
DONOR HH12 NH1
DONOR HH21 NH2
DONOR HH22 NH2
ACCEPTOR O C
IC -C   CA   *N   HN    1.3496 122.4500  180.0000 116.6700  0.9973
IC -C   N    CA   C     1.3496 122.4500  180.0000 109.8600  1.5227
IC N    CA   C    +N    1.4544 109.8600  180.0000 117.1200  1.3511
IC +N   CA   *C   O     1.3511 117.1200  180.0000 121.4000  1.2271
IC CA   C    +N   +CA   1.5227 117.1200  180.0000 124.6700  1.4565
IC N    C    *CA  CB    1.4544 109.8600  123.6400 112.2600  1.5552
IC N    C    *CA  HA    1.4544 109.8600 -117.9300 106.6100  1.0836
IC N    CA   CB   CG    1.4544 110.7000  180.0000 115.9500  1.5475
IC CG   CA   *CB  HB1   1.5475 115.9500  120.0500 106.4000  1.1163
IC CG   CA   *CB  HB2   1.5475 115.9500 -125.8100 109.5500  1.1124
IC CA   CB   CG   CD    1.5552 115.9500  180.0000 114.0100  1.5384
IC CD   CB   *CG  HG1   1.5384 114.0100  125.2000 108.5500  1.1121
IC CD   CB   *CG  HG2   1.5384 114.0100 -120.3000 108.9600  1.1143
IC CB   CG   CD   NE    1.5475 114.0100  180.0000 107.0900  1.5034
IC NE   CG   *CD  HD1   1.5034 107.0900  120.6900 109.4100  1.1143
IC NE   CG   *CD  HD2   1.5034 107.0900 -119.0400 111.5200  1.1150
IC CG   CD   NE   CZ    1.5384 107.0900  180.0000 123.0500  1.3401
IC CZ   CD   *NE  HE    1.3401 123.0500  180.0000 113.1400  1.0065
IC CD   NE   CZ   NH1   1.5034 123.0500  180.0000 118.0600  1.3311
IC NE   CZ   NH1  HH11  1.3401 118.0600 -178.2800 120.6100  0.9903
IC HH11 CZ   *NH1 HH12  0.9903 120.6100  171.1900 116.2900  1.0023
IC NH1  NE   *CZ  NH2   1.3311 118.0600  178.6400 122.1400  1.3292
IC NE   CZ   NH2  HH21  1.3401 122.1400 -174.1400 119.9100  0.9899
IC HH21 CZ   *NH2 HH22  0.9899 119.9100  166.1600 116.8800  0.9914
```

Reference Residues

```
RESI THR          0.00  !   Threonine
GROUP
ATOM N    NH1    -0.47  !      |
ATOM HN   H       0.31  !   HN-N
ATOM CA   CT1     0.07  !      |      OG1--HG1
ATOM HA   HB      0.09  !      |     /
GROUP                   !   HA-CA--CB-HB
ATOM CB   CT1     0.14  !      |     \
ATOM HB   HA      0.09  !      |      CG2--HG21
ATOM OG1  OH1    -0.66  !    O=C     / \
ATOM HG1  H       0.43  !      | HG21 HG22
GROUP
ATOM CG2  CT3    -0.27
ATOM HG21 HA      0.09
ATOM HG22 HA      0.09
ATOM HG23 HA      0.09
GROUP
ATOM C    C       0.51
ATOM O    O      -0.51
BOND CB CA  OG1 CB   CG2 CB    N   HN
BOND N  CA    C   CA    C   +N    CA  HA
BOND CB HB  OG1 HG1  CG2 HG21  CG2 HG22  CG2 HG23
DOUBLE  O   C
IMPR N -C CA HN  C CA +N O
DONOR HN N
DONOR HG1 OG1
ACCEPTOR OG1
ACCEPTOR O C
IC -C   CA   *N   HN    1.3471 124.1200  180.0000 114.2600  0.9995
IC -C   N    CA   C     1.3471 124.1200  180.0000 106.0900  1.5162
IC N    CA   C    +N    1.4607 106.0900  180.0000 117.6900  1.3449
IC +N   CA   *C   O     1.3449 117.6900  180.0000 120.3000  1.2294
IC CA   C    +N   +CA   1.5162 117.6900  180.0000 124.6600  1.4525
IC N    C    *CA  CB    1.4607 106.0900  126.4600 112.7400  1.5693
IC N    C    *CA  HA    1.4607 106.0900 -114.9200 106.5300  1.0817
IC N    CA   CB   OG1   1.4607 114.8100  180.0000 112.1600  1.4252
IC OG1  CA   *CB  HB    1.4252 112.1600  116.3900 106.1100  1.1174
IC OG1  CA   *CB  CG2   1.4252 112.1600 -124.1300 115.9100  1.5324
IC CA   CB   OG1  HG1   1.5693 112.1600 -179.2800 105.4500  0.9633
IC CA   CB   CG2  HG21  1.5693 115.9100 -173.6500 110.8500  1.1104
IC HG21 CB   *CG2 HG22  1.1104 110.8500  119.5100 110.4100  1.1109
IC HG21 CB   *CG2 HG23  1.1104 110.8500 -120.3900 111.1100  1.1113
```

```
RESI TYR          0.00
GROUP
ATOM N    NH1    -0.47  !      |         HD1  HE1
ATOM HN   H       0.31  !   HN-N          |    |
ATOM CA   CT1     0.07  !      |    HB1  CD1--CE1
ATOM HA   HB      0.09  !      |    |   //      \\
GROUP                   !   HA-CA--CB--CG      CZ--OH
ATOM CB   CT2    -0.18  !      |    |    \  __  /     \
ATOM HB1  HA      0.09  !      |    HB2  CD2--CE2     HH
ATOM HB2  HA      0.09  !    O=C           |    |
GROUP                   !      |         HD2  HE2
ATOM CG   CA      0.00
GROUP
ATOM CD1  CA     -0.115
ATOM HD1  HP      0.115
GROUP
ATOM CE1  CA     -0.115
ATOM HE1  HP      0.115
GROUP
ATOM CZ   CA      0.11
ATOM OH   OH1    -0.54
ATOM HH   H       0.43
GROUP
ATOM CD2  CA     -0.115
ATOM HD2  HP      0.115
GROUP
ATOM CE2  CA     -0.115
ATOM HE2  HP      0.115
GROUP
ATOM C    C       0.51
ATOM O    O      -0.51
BOND CB  CA   CG  CB   CD2 CG   CE1 CD1
BOND CZ  CE2  OH  CZ
BOND N   HN   N   CA    C   CA   C   +N
BOND CA  HA   CB  HB1  CB  HB2  CD1 HD1  CD2 HD2
BOND CE1 HE1  CE2 HE2  OH  HH
DOUBLE   O   C   CD1 CG  CE1  CZ  CE2 CD2
IMPR N -C CA HN  C CA +N O
DONOR HN N
DONOR HH OH
ACCEPTOR OH
ACCEPTOR O C
IC -C   CA   *N   HN    1.3476 123.8100  180.0000 114.5400  0.9986
IC -C   N    CA   C     1.3476 123.8100  180.0000 106.5200  1.5232
IC N    CA   C    +N    1.4501 106.5200  180.0000 117.3300  1.3484
IC +N   CA   *C   O     1.3484 117.3300  180.0000 120.6700  1.2287
IC CA   C    +N   +CA   1.5232 117.3300  180.0000 124.3100  1.4513
IC N    C    *CA  CB    1.4501 106.5200  122.2700 112.3400  1.5606
IC N    C    *CA  HA    1.4501 106.5200 -116.0400 107.1500  1.0833
IC N    CA   CB   CG    1.4501 111.4300  180.0000 112.9400  1.5113
IC CG   CA   *CB  HB1   1.5113 112.9400  118.8900 109.1200  1.1119
IC CG   CA   *CB  HB2   1.5113 112.9400 -123.3600 110.7000  1.1115
```

```
IC CA   CB   CG   CD1   1.5606 112.9400   90.0000 120.4900  1.4064
IC CD1  CB   *CG  CD2   1.4064 120.4900 -176.4600 120.4600  1.4068
IC CB   CG   CD1  CE1   1.5113 120.4900 -175.4900 120.4000  1.4026
IC CE1  CG   *CD1 HD1   1.4026 120.4000  178.9400 119.8000  1.0814
IC CB   CG   CD2  CE2   1.5113 120.4600  175.3200 120.5600  1.4022
IC CE2  CG   *CD2 HD2   1.4022 120.5600 -177.5700 119.9800  1.0813
IC CG   CD1  CE1  CZ    1.4064 120.4000   -0.1900 120.0900  1.3978
IC CZ   CD1  *CE1 HE1   1.3978 120.0900  179.6400 120.5800  1.0799
IC CZ   CD2  *CE2 HE2   1.3979 119.9200 -178.6900 119.7600  1.0798
IC CE1  CE2  *CZ  OH    1.3978 120.0500 -178.9800 120.2500  1.4063
IC CE1  CZ   OH   HH    1.3978 119.6800  175.4500 107.4700  0.9594

RESI PTYR        -1.00
GROUP
ATOM N    NH1    -0.47  !      |         HD1   HE1
ATOM HN   H       0.31  !   HN-N          |     |
ATOM CA   CT1     0.07  !      |    HB1   CD1--CE1        OC1
ATOM HA   HB      0.09  !      |    |    //      \\        |
GROUP                   !   HA-CA--CB--CG       CZ--OH--PO4—OC2
ATOM CB   CT2    -0.18  !      |    |    \   __   /         |
ATOM HB1  HA      0.09  !      |    HB2   CD2--CE2        OC3
ATOM HB2  HA      0.09  !    O=C           |     |
GROUP                   !      |         HD2   HE2
ATOM CG   CA      0.00
GROUP
ATOM CD1  CA     -0.115
ATOM HD1  HP      0.115
GROUP
ATOM CE1  CA     -0.115
ATOM HE1  HP      0.115
GROUP
ATOM CZ   CA      0.11
ATOM OH   OH1    -0.54
ATOM PO4  P04     1.098
ATOM OC1  OC     -0.556
ATOM OC2  OC     -0.556
ATOM OC3  OC     -0.556
GROUP
ATOM CD2  CA     -0.115
ATOM HD2  HP      0.115
GROUP
ATOM CE2  CA     -0.115
ATOM HE2  HP      0.115
GROUP
ATOM C    C       0.51
ATOM O    O      -0.51
BOND CB   CA   CG   CB   CD2 CG   CE1 CD1
BOND CZ   CE2  OH   CZ
BOND N    HN   N    CA    C   CA   C   +N
BOND CA   HA   CB   HB1  CB  HB2  CD1 HD1  CD2 HD2
BOND CE1  HE1  CE2  HE2  OH  PO4  PO4 OC1  PO4 OC2
BOND PO4  OC3
```

```
DOUBLE    O    C    CD1 CG   CE1   CZ   CE2 CD2
IMPR N -C CA HN   C CA +N O
DONOR HN N
ACCEPTOR OH
ACCEPTOR O C
IC -C    CA   *N    HN    1.3476 123.8100  180.0000 114.5400  0.9986
IC -C    N    CA    C     1.3476 123.8100  180.0000 106.5200  1.5232
IC N     CA   C     +N    1.4501 106.5200  180.0000 117.3300  1.3484
IC +N    CA   *C    O     1.3484 117.3300  180.0000 120.6700  1.2287
IC CA    C    +N    +CA   1.5232 117.3300  180.0000 124.3100  1.4513
IC N     C    *CA   CB    1.4501 106.5200  122.2700 112.3400  1.5606
IC N     C    *CA   HA    1.4501 106.5200 -116.0400 107.1500  1.0833
IC N     CA   CB    CG    1.4501 111.4300  180.0000 112.9400  1.5113
IC CG    CA   *CB   HB1   1.5113 112.9400  118.8900 109.1200  1.1119
IC CG    CA   *CB   HB2   1.5113 112.9400 -123.3600 110.7000  1.1115
IC CA    CB   CG    CD1   1.5606 112.9400   90.0000 120.4900  1.4064
IC CD1   CB   *CG   CD2   1.4064 120.4900 -176.4600 120.4600  1.4068
IC CB    CG   CD1   CE1   1.5113 120.4900 -175.4900 120.4000  1.4026
IC CE1   CG   *CD1  HD1   1.4026 120.4000  178.9400 119.8000  1.0814
IC CB    CG   CD2   CE2   1.5113 120.4600  175.3200 120.5600  1.4022
IC CE2   CG   *CD2  HD2   1.4022 120.5600 -177.5700 119.9800  1.0813
IC CG    CD1  CE1   CZ    1.4064 120.4000   -0.1900 120.0900  1.3978
IC CZ    CD1  *CE1  HE1   1.3978 120.0900  179.6400 120.5800  1.0799
IC CZ    CD2  *CE2  HE2   1.3979 119.9200 -178.6900 119.7600  1.0798
IC CE1   CE2  *CZ   OH    1.3978 120.0500 -178.9800 120.2500  1.4063
IC CE1   CZ   OH    HH    1.3978 119.6800  175.4500 107.4700  0.9594
(And other ICs which you should figure out.)
```

LAB 4: ENERGY MINIMIZATION

In the previous labs we have, on a number of occasions, asked you to perform an energy minimization without going into detail about the process. In this lab we will explore minimization and the Energy function, but first, let's learn more about the next level in the CHARMM data structures: the PSF file.

I. The Principle Structure File and Patches

The PSF file was originally called the Protein Structure File, but the name was converted to the Principle Structure File with the advent of DNA, lipids, etc. It is like the residue topology file, but contains the topology information for your whole system (NAMD PSF tutorial). For instance, if you have a 6-residue peptide, the PSF will contain the atoms types and partial charges for each of the residues, as well as the connections between the residues. In this regard, the PSF is a super-RTF. However, the style of the bond-pair and other bonded interactions are a bit different from the RTF and there are additional structures for H-bonding and other purposes that are non-transparent at the end. We will not concern ourselves with these, but will examine the PSF to learn exactly what takes place when a PSF patch is applied.

PSF patches are designed to modify the PSF for a segment after it has been generated. For instance, you might want to attach a palmitoyl group to a Cys side chain, or link a Heme residue to a His side chain after building a protein. These "post-translational modifications" are carried out in CHARMM by commands that delete existing atoms in the protein and then add in others.

A **patch** is very similar to a residue (struct.doc). Both are defined in the RTF. Residues are specified with the RESI keyword (short for residue) while patches are specified by the PRES keyword (short for patch residue). Patches are usually used to modify existing PSF structures, while residues are used to generate PSF structures. For example, PRES DISU removes the terminal H on two Cys side chains and then installs a bond between the side chain Sulfurs:

```
PRES DISU         -0.36      ! patch for disulfides. Patch must be
1-CYS and 2-CYS.
GROUP                        ! use in a patch statement
ATOM 1CB  CT2     -0.10      !
ATOM 1SG  SM      -0.08      !              2SG--2CB--
GROUP                        !                        /
ATOM 2SG  SM      -0.08      ! -1CB--1SG
ATOM 2CB  CT2     -0.10      !
DELETE ATOM 1HG1
DELETE ATOM 2HG1
BOND 1SG 2SG
ANGLE 1CB 1SG 2SG 1SG 2SG 2CB
DIHE 1HB1 1CB 1SG 2SG 1HB2 1CB 1SG 2SG
DIHE 2HB1 2CB 2SG 1SG 2HB2 2CB 2SG 1SG
DIHE 1CA 1CB 1SG 2SG 1SG 2SG 2CB 2CA
DIHE 1CB 1SG 2SG 2CB
!DIHE 1CB 1SG 2SG 2CB
IC 1CA  1CB  1SG  2SG   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1CB  1SG  2SG  2CB   0.0000  0.0000   90.0000  0.0000  0.0000
```

```
IC 1SG  2SG  2CB  2CA   0.0000  0.0000  180.0000  0.0000  0.0000
```

Notice where the two hydrogen atoms are deleted and the disulfide bond is added. To apply the patch, you issue the command after the segment(s) containing the protein(s) is (are) generated:

```
PATCH DISU segn1 resi1 segn2 resi2
```

Segn1 and resi1 represent the name of the segment and the number of the residue that correspond to the first residue in the patch. Segn2 and resi2 likewise represent the name of the segment and the number of the second residue in the patch. They are treated as arguments and used to replace 1-CYS and 2-CYS as they appear in the text.

Let's use this with a 16-residue fragment from SNAP25, the synaptic vesicle fusion protein, which has 4 Cys residues in the center. One might imagine that these would be prone to be reduced when the protein is in the cytoplasm, allowing the protein to stretch out, and then to be oxidized when the protein is in the membrane, inducing a hairpin conformation for the protein.

The fragment sequence for SNAP25 can be read into CHARMM using the stream file "build_snap25.str." Open "build_snap25.str" and observe that this script opens the SNAP25 sequence, generates the segment "SNAP," and builds the Cartesian coordinates from the IC table (note: you should be able to do this on your own. If you are confused about what is going on in this file, you should review the previous labs so that you have a good understanding of the basics of building proteins).

In a new input file, open the topology and parameter files and then stream "build_snap25.str." After streaming "build_snap25.str", use PRINT PSF to write the PSF data structure to the log file at this juncture. Then, add the PATCH command shown above twice to introduce two disulfide bonds, one between residues 5 and 12 and another between residues 8 and 10. Then PRINT PSF again. Write a coordinate file in PDB format for viewing the structure, which should, at this stage, just be an extended rod. Also, write the PSF and coordinates to .PSF and .CRD files for use in the next script. That way you won't have to rebuild the molecules again. The CHARMM commands to write these files are as follows:

```
OPEN WRITE UNIT 1 CARD NAME snap25.pdb
WRITE COOR PDB UNIT 1 CARD
* SNAP25 FRAGMENT
*
CLOSE UNIT 1

OPEN WRITE UNIT 1 CARD NAME snap25.crd
WRITE COOR UNIT 1 CARD
* SNAP25 FRAGMENT
*
CLOSE UNIT 1

OPEN WRITE UNIT 2 CARD NAME snap25.psf
WRITE PSF UNIT 2 CARD
* SNAP25 FRAGMENT
*
CLOSE UNIT 2
```

Question 1: How has the PSF changed? Why doesn't VMD show the disulfide bonds? (While it isn't necessary, you might find it helpful to write the PSF's to individual files that can be loaded in VMD and compared, rather than just viewed in the log file.)

If we were building the whole protein, we might next wind the appropriate residues at each end of the protein into alpha-helices using the stream files from last week. These files change the values of the dihedrals PSI and PHI in the IC table, but do not change the Cartesian coordinates. To change the Cartesian coordinates, you would have to first initialize the Cartesian coordinates of those atoms you expect to be moved in the new conformations. For this, you would use the COOR INIT command, selecting those atoms to be initialized. IC BUILD could then rebuild the initialized coordinates based upon the remaining coordinates and the edited IC table. We will skip over this, however, for the sake of time so that we can move on to minimization.

II. Minimization and Energy

Now we need to allow the two halves of the fragment to form the hairpin and the disulfide bonds to shorten to a normal length. Energy minimization is an algorithm that searches for a local minimum (minimiz.doc). Molecular dynamics can be used to further anneal the structure and explore conformational space. Use the PSF and CRD files that you generated together with the stream file below named iama_minimzer1.str (which is located in the charmmlab directory). This will minimize the structure for 500 steps of **steepest decent (SD)** writing the energy out to the log file every 10 steps. Run this STR file and note, in the log file produced, what the energy was after 0, 10, and 500 steps. Particularly, pay attention to the bond energy term, which should be huge to begin with. You will see in the LOG file that CHARMM is warning you that certain pairs of atom are too close together or that certain dihedrals are stretched. These warnings should decrease in number as the minimization progresses.

Next, alter the .str file so that it now does 500 steps of Adopted Basis Newton - Raphson (ABNR) minimization on the initial structure instead and save the STR file as iama_minimizer2.str. Note the values after 0, 10 and 500 steps. ABNR is a somewhat more complicated algorithm that is designed to find the minimum faster, albeit less robustly, than SD. It is usually used to refine a fit after SD is used to get close to a local minimum. The structure of the two files is shown below.

```
**********************iama_minimizer1.str*****************************
*  Minimizes the peptide using SD minimization
*

STREAM TOPPARM.STR

OPEN READ CARD UNIT 11 NAME snap25.psf
READ PSF CARD UNIT 11
CLOSE UNIT 11

OPEN READ CARD UNIT 11 NAME snap25.crd
READ COOR CARD UNIT 11
CLOSE UNIT 11

MINI SD NSTEP 500 NPRINT 10
STOP
```

```
**********************iama_minimizer2.str**************************
*   Minimizes the peptide using ABNR minimization
*

STREAM TOPPARM.STR

OPEN READ CARD UNIT 11 NAME snap25.psf
READ PSF CARD UNIT 11
CLOSE UNIT 11

OPEN READ CARD UNIT 11 NAME snap25.crd
READ COOR CARD UNIT 11
CLOSE UNIT 11

MINI ABNR NSTEP 500 NPRINT 10
STOP
```

Question 2: Which method has reached a lower value of energy after 500 steps (SD or ABNR)? By how much?

ABNR sure seems to be slower so far, but it is designed to do the best when the total energy is not at exponential values. Create two new stream files: one that does 100 SD on the original structure, and one that does 50 SD then 50 ABNR. Run each of these stream files.

Question 3: Which structure has the lower energy now? By how much?

Conjugate Gradient (CONJ) is supposed to be very similar to SD (each step takes about the same amount of time) except that it has a slightly better algorithm - one that remembers how effective the last minimization step is and directs the step orientation and it's size accordingly. Write a script that minimizes the original structure for 50 steps of CONJ.

Question 4: How did the 50 steps of CONJ compare to 50 steps of SD? (Retrieve the 50th step of SD from the log file – located part way through the 500-step minimization).

Now write a stream file that minimizes the original structure first for 50 SD, then 1000 ABNR. Note that the structure drops in energy by smaller and smaller amounts. Also note the rms force per atom or root-mean-squared gradient, which is listed in the energy print out as GRMS. The rms force is the root mean squared force on the atoms of this structure - a measure of the stress on the molecule. At a local minimum, the force would be equal to 0.0. It is usually a good indication of how well minimized a structure is.

How good is good enough? Usually an rms force less than 1.0 is the minimum preparation needed for heating in molecular dynamics. A value less than 0.1 is considered very well minimized. Remember – even an extremely well minimized structure is probably still just in a local minimum, so no use getting carried away here. A value less than 1.0 is generally good enough that the system will not spontaneously heat up and cause an "Energy Tolerance Exceeded" crash in the initial stages of molecular dynamics.

By the way, when performing a minimization, CHARMM looks at the change in energy of the structure. If the change is smaller than a set tolerance parameter, the program decides that the structure is minimized well enough and quits out of the minimization. You may see this some day and wonder why CHARMM stopped in the middle of a minimization. It is not a bug--this is CHARMM's way of being more efficient.

III. Constraints

CHARMM allows you to add energy terms to minimizations (and dynamics) called **constraints** (cons.doc). There are three basic types of constraints: Atom, Distance, and Dihedral. Harmonic atom constraints allow you to confine an atom near its current X Y Z coordinates by a restoring force. The strength of the force can be varied. The associated energy has units of kcal/ (Angstrom from the initial position)2. Another type of atom constraint is the Fixed constraint, which toggles off the "move flag" for selected atoms. They will not be moved during minimization or dynamics, and interactions between fixed atoms will not be added to the potential energy. Distance constraints allow you to fix the distance between two atoms with a specific force, allowing the user to keep specific atoms within a certain distance of one another. Dihedral constraints create a restoring force which holds the value of a dihedral angle near to a specific value. These commands each allow you to control a minimization or a dynamics run in specific ways. Here we will use constraints to answer a real question from our research:

Years ago we were trying to design a cyclic peptide to bind calcium. The sequence being explored was: EPPEP PEPPE PP. We hoped that the 4 negatively charged glutamic acids would bind calcium, and the prolines would keep the peptide well defined. Experimentally, the peptide would be synthesized first as a linear peptide; then the N- and C-termini would be linked chemically forming a cyclic peptide. But could this cyclization step be energetically feasible in the test tube? It is possible that the peptide might stay extended, that the N- and C-termini could only approach each other under unreasonable strain, and that such contacts would happen at so low a rate that the reaction would not produce much product. Or it might easily bend, and we could expect quite a lot of product. This is something we should try to estimate before blowing $1000 on synthesizing this thing. So let's use CHARMM to find out the stress involved in cyclizing the peptide in a simple way.

First use CHARMM to create the linear version of the peptide. Write a .pdb file named "pre-min.pdb" for viewing with VMD. Minimize it for 200 ABNR. Write another .pdb file named "post-min.pdb." Finally, write .crd and .psf files so that you don't have to rebuild and minimize the structure in the next section. Compare the two .pdb files in VMD. The N- & C-termini don't rush toward each other, but the structure does seem to change a little.

Minimization just gives us a local minimum. What we wish to see may not be a minimum, but may not be too far from one in energy. If we did dynamics for long enough we could watch for a conformation with the N- & C-termini near one another, but this might take quite a long dynamics run. Instead we will use a NOE constraint, which is a "distance" or "target" constraint. (NOE stands for Nuclear Overhauser Effect, which is a proximity signal obtained in NMR).

Create a new stream file named "constrained.str." Copy the format of "iama_minimizer1.str" to load the structure that you just made above. (Be sure to only use "iama_minimizer1.str" as a template, i.e. do not include any of the commands (like MINI for example), other than those necessary to load the protein.)

With the protein loaded, you can apply the constraint. The following code will use an NOE constraint to pull the ends together:

```
NOE
  ASSI SELE ATOM SEGN 1 N END SELE ATOM SEGN 12 C END -
  KMAX 100.0 RMAX 3.0 FMAX 1000.0 RSUM
END
```

The NOE keyword marks the beginning of an NOE block and the keyword END marks the end. ASSI stands for assign. SELE invokes the selection command that was covered in the Molecular Structure Manipulation lab. SEGN should be changed to the name of the segment that you generated when you created the peptide. The space followed by a hyphen means that the next line should be concatenated onto the current line. This code will assign a constraint between the N atom of the first residue and the C atom of the twelfth residue. RMAX specifies the largest distance between the two selections that will be penalty free. For greater distances, a harmonic penalty will be applied with KMAX as the energy constant.

Calculate the energy of the structure, using the ENER command. Now do 25 steps of SD and 1000 steps of ABNR. Write a .pdb file named "constrained.pdb" to view in VMD. Turn off the constraints using the command RESET (which stands for reset) embedded in a new NOE block. Then recalculate the energy. Run your script and open "constrained.pdb" in VMD. Note that the ends came together. This is similar to what happened when we cyclized the SNAP25 fragment with the disulfide bond. Open the log file and note that the energy of the constraint is also calculated after the first ENER command. Compare the energies with the constraint turned on and with the constraint turned off.

Question 5: What is the difference in energy between the linear peptide and the constrained peptide? Do you think that the structure can fold in this way, or does the energy difference select against it? Another way to look at the problem is to minimize now that the constraints are released and see if the ends fly apart or if they are stable near one another. Try it. What happens?

Question 6: There are a number of problems with doing the calculation in this way. I can think of at least 3 major ones. Can you come up with two of them? Put down as many as you can think of. Some are not at all obvious.

Now go back and use CONS FIX SELE ATOM SEGN 1 N END (again, replace SEGN with the name of your segment) (corman.doc) command to fix the N-terminus amino acid in place in addition to the NOE distance constraint. This time only the C-terminus should move. A word of warning - if the line connecting two atoms which you wish to distance-constrain passes through another group of atoms the minimization could get real ugly fast as the atoms crash into one another. In this case it might be necessary to turn off VDW interactions or change the initial conformation by editing dihedrals and rebuilding.

LAB 5: INTRODUCTION TO DYNAMICS

This lab will serve as basic walkthrough of a dynamics simulation. The objectives for this lab are as follows:

- Build a system, including building a water box, neutralization, and configuration of the ligand.
- Heat a system in preparation for a dynamics simulation.
- Perform a short dynamics simulation
- Analyze the simulations with simple analysis tools in VMD.

The molecular dynamics simulationd you will perform will allow you to analyze of the effect of an electric field on the protein configuration of the Luciferase enzyme. Luciferase is an enzyme that produces bioluminescence in fireflies through converting luciferin and ATP into oxyluciferin and AMP, releasing a photon of light as oxyluciferin returns to ground state. The luciferase that we will be analyzing is found on PDB.org under the identifier 2D1S. Its crystal structure contains a luciferyl-AMP intermediate within its active site.

Dr. Brian Mazzeo at BYU has conducted several experiments where he analyzed the effect of an electric field on luciferase fluorescence. His experiments are currently inconclusive, but dynamics simulations could certainly offer further insight. The dynamics simulations you perform in this lab should demonstrate how the protein and ligand react to a strong electric field.

I. Molecular Dynamics Background

Energy minimizations are designed to find the energy wells for static molecules based only on the potential energy. **Molecular dynamics** allow you to also look at changes that increase the entropy of an entire system, and provides a much more flexible search of conformational space. We will further explore entropy in the lab on perturbational techniques later in the course.

Dynamics runs take time. In research, we commonly write CHARMM scripts that do the calculation for us and leave them running for days at a time. This lab will not take that long. It is important that you look over the example stream file and understand it, but not that you understand all of the variables set in the dynamics command. Most of them are never changed from script to script. Rather, they are just copied to the new dynamics script. In fact, many have found that the easiest way of writing a CHARMM script is to use an old one as a template. That is what is expected for you to do here. If you get confused try referring to the dynamics command entry in the CHARMM manual or online documentation.

Dynamics simulations have a few common outputs including the following file types:

- .DCD: These are the files with the Dynamics CoorDinates. They tend to take up a lot of memory.
- .RST: These are the ReSTart files. They contain info (coordinates and velocities) for continuing the dynamics from this point.
- .CRD: Dynamics runs save the final CooRDinates of each sub-section to CRD files.

The dynamics run is typically broken down into three phases (actually three separate dynamics simulations): Heating, Equilibration: and Dynamics. It starts out as heating - the temperature is increased, in this case, from 0 K to 300 K in 3000 steps of dynamics. In this phase, a random velocity is assigned to each of the atoms in the molecule corresponding to the current temperature. This temperature is increased incrementally as the run progresses. You are not limited to 300 K. You could heat to 1000 K if you felt like it. Some people do this as a way to search areas of conformational spaces that are separated by high energy barriers. One could also take a molecule at 1000K and 'heat' it with negative temperature increments back down to 300 K. This is called "simulated annealing" and is similar to minimization. We will stick with heating for the moment.

Equilibration comes next. Why is this needed? As you give atoms kinetic energy, the bonds will stretch and compress, angles will bend back and forth, etc. If we gave the molecule kinetic energy and let it go, we would find that some of this energy will actually become potential energy. This is exactly what physicists are talking about when they mention the thermodynamic principle of energy equipartition. In our case the net effect is that energy goes sloshing back and forth from kinetic to potential causing the temperature to fluctuate for the molecule. When the conformation or configuration of the system changes, the potential energy changes causing a shift in the kinetic energy, and hence, the temperature. To correct for this you will run an equilibration run. This periodically measures the temperature and scales the velocity back to the equilibration temperature (300K). After a while, this will get the large fluctuations to decrease, but you cannot ever get them to go below a certain level. This is normal and even realistic, as temperature is an average quantity anyway.

Finally, you do the actual dynamics simulation itself. The main drawback of molecular dynamics is the amount of time required to perform simulations. Most reactions take place on the microsecond or millisecond time scale, which requires a substantial amount of computing power and resources to effectively simulate for most applications.

II. Building the System

A crucial step in performing dynamics simulations is to properly set up your system. This includes solvation (surrounding the protein in generated water molecules), neutralization (adding ions to give the system a net charge of 0), and performing energy minimizations. The complexity of this process depends on the complexity of the dynamics simulation you wish to perform.

Prepare the Protein and Ligand

The combine_protein_lig.inp script inside the directory 1_put_ligand_in_protein is a simple input script that combines the protein and the ligand in one coordinate file and performs some minimizations on the system. Open the script and answer the following question:

> *Question 1: Describe this script. What are its inputs? What is being manipulated and why? What is the output?*

Make sure the submission script's input and output files are designated correctly and execute the submit script. The output is a CRD and PSF that contain the protein with the ligand in the active site. Feel free to view the output in VMD.

Solvation

This is a very simplified solvation script. The 100x100 angstrom water box has already been generated by www.charmm-gui.org. You can generate your own water box in CHARMM, but using the CHARMM-GUI can speed things up a bit. Make sure the solvation script refers to the psf and crd you generated previously. Open solvator.inp and take a look at the script. You'll notice a deletion command that is written as follows:

```
delete atom sort -
select .byres. (resn TIP3 .AND. type oh2 .and. -
((.not. (resn TIP3 .OR. hydrogen)) .around. 2.8)) end
```

Question 2: What is this deletion doing? Why is this important?

Execute the submission script to solvate the molecule.

Question 3: How many water molecules did you generate?

Neutralization

Dr. Mazzeo performed his experiments in a solution with 0.2M ammonium sulfate. For the purpose of our experiment, we'll be using potassium chloride. Using www.charmm-gui.org, it was determined that 108 KCl molecules will result in a 0.2M concentration. However, the protein carries a net charge of +1, so a total of 109 Cl^- ions are necessary to neutralize the system.

Run the neutralize.inp script. This script streams 3 additional subscripts: watervars.str, ioncoor.str, and watercolor.str. These three scripts identify 217 water molecules to delete through randomly generated numbers, making room for the ions to be added. The water ions are then deleted, and the ions replace the waters' former positions with 2 exceptions: 2 Cl^- ions are added within the protein to take the place of chloride ions that were found in the original pdb script.

III. Dynamics

Heating

The system must be heated to room temperature prior to simulation. Open the folder 4_minimize_heat in WinSCP and open the submit file batch.submit. This file submits your script to the Fulton supercomputer. The line **`#SBATCH -N1 -n16 --mem-per-cpu=2G -t10:00:00 -C 'm7'`** designates several important settings for your submission. **`-N1`** refers to the number of nodes that you'll run your script on. Generally speaking, the more nodes you request the faster your script will run. **`-n16`** refers to the total number of processors requested across all nodes. **`-mem-per-cpu`** requests the amount of RAM per CPU, which is usually kept at 1G or 2G. **`-t10:00:00`** designates the amount of time you anticipate your script to run. **`-C 'm7'`** designates that we'll be running our scripts on the m7 partition of the supercomputer (see sbatch documentation).

Open the input file minimize_heat.inp. Scroll down to the "! CRYSTAL" portion of the script. This command generates an infinite crystal of our system. This will allow the system to remain confined to a cubic structure, and the outer edges of the cube will factor in the Van der Waals of the molecules that are within 14 angstroms of the outer edge.

Question 4: What do you think "crystal defi cubi 100.0 100.0 100.0 90.0 90.0 90.0" is designating?

View the "! CONSTRAINTS" portion of the script. Three constraints will be applied in this script. We'll initially apply a harmonic constraint on the protein, to ease the impact of the electric field. **`PULL EFIELD 1E9 XDIR 1.0 SELE ALL END`** (cons.doc) creates our 1E9-volt electric field constraint in the X-direction, which is applied to the whole system. Finally, we must apply a center-of-mass constraint to the protein, which will prevent the charged luciferase enzyme from migrating through the water box as a result of the electric field.

View the "! MINIMIZATION" portion of the script.

Question 5: What kind of minimizations are being performed and how many steps of each?

View the "! HEAT SYSTEM TO 300" segment of the script. You can see that two files are opened for writing: 0.rst and 0.dcd. After these files are opened, our heating script follows:

```
dynamics cpt leap verl strt nstep 40000 time 0.0015 -
iunrea -1 iunwri 31 iuncrd 32 -
isvfrq 2000 nsavc 200 nsavv 0 -
inbfrq -1 nprint 250 iprfrq 0 ntrfrq 100 -
pcon pgam 25 pmass 500 pref 1.0 surface tension 0.0 -
ihtfrq 2 teminc 0.06 -
imgfrq 50 ixtfrq 1000 cutim 14 -
iasors 0 iasvel 1 iscvel 1 iseed 1 -
firstt 0.0 finalt 300.0
```

There are several values we need to understand when running this heating script (dynamc.doc):

- **`dynamics cpt leap verl strt`** – this begins our dynamics simulation as a constant pressure and temperature (cpt) leap verlet simulation.
- **`nstep 40000`** – this designates the number of steps to be performed.
- **`time 0.0015`** – this designates the length in time (in picoseconds) of a single step, or how long a simulation is allowed to move before the velocities of the system are recalculated and adjusted. .0015 picoseconds, or 1.5 femtoseconds.

Question 6: How many picoseconds of simulation would be carried out with the current script?

- **`nsavc 200`** – this determines the number of steps before the coordinates are saved to the .dcd file
- **`iunwri 31`** – this will write our RESTART file to the open unit 31. Restart files are essential for restarting our simulations from where our previous simulation finished off.
- **`iuncrd 32`** – this will write our .dcd file to the open unit 32.
- **`pcon`** – this command and its subsequent values designate that the simulation will run under constant pressure.
- **`firstt 0.0 finalt 300.0`** – this will heat our script from 0.0 K to 300.0 K.

Now run the script. Submit your script to the supercomputer in the UNIX console by typing **`sbatch batch.submit`**.

You can view the queue status of your simulation by typing in the command line **`watch squeue -u [username]`**. This script will take several hours to complete, if you do not make any changes to the submit script.

Equilibration

We are now going to start the equilibration phase. Open the folder 5_simulate, and the input file 1E9_sim. You'll notice that this file is very similar to the heating script, without the minimizations. The differences are primarily in the dynamics simulation. Notice the differences in the dynamics settings:

```
dyna cpt leap restart nstep 100000 timestep 0.0015 -
nprint 100 iprfrq 5000 ntrfrq 5000 -
iunrea 30 iunwri 31 iuncrd 32 -
nsavc 1000 nsavv 0 -
imgfrq 50 ixtfrq 1000 cutim 14 -
pcons pint pref 1.0 pmass 500 pgamma 0 -
hoov reft 300 tmass 2000.0 tbath 300 firstt 300 finalt 300
```

Question 7: What are the differences between this simulation and the heating script? Ask your TA what these differences mean, or look them up on www.charmm.org.

You'll notice that there are various folders within 5_simulate. This is necessary to organize the output files of our simulation. To perform a long simulation we have to pause the simulation and restart it from where we left off. This is to decrease the amount of consecutive wall time on the supercomputer, and to protect us from having to restart the entire simulation should an error occur.

Now prepare the submission script for equilibration. If you want to perform this simulation yourself, the equilibration should take a few days to complete, depending on the resources you request. Luckily, we have the finished product already available to you in the directory charmmlab/lab_files/luciferase_output.

IV. Analysis

We will use VMD to perform an analysis of the Root-Mean-Square Deviation (**RMSD**), as well as graph the distance between two residues over time.

RMSD Analysis

In WinSCP, transfer the system_min.psf, 1.crd, and 1.dcd files onto your local computer. When complete, open VMD on your computer. Go to File > New Molecule. Load 1.crd, be sure to specify that the file contains CHARMM coordinates in the drop down menu. Load system_min.psf and 1.dcd.

Once all the frames are loaded, go to Extensions > Analysis > RMSD Visualizer Tool. In the new window, type "all segid PROA" in the atom selection dialogue box. Next, click "ALIGN" and then "RMSD". Select "Plot Result."

Question 8: Describe the data that you see. What causes the changes in RMSD?

Distance between Atoms over Time

This is another easy analysis that graphs the distance between atoms over time. Create a bond in VMD between any atom in the ligand and an atom near it in the surrounding luciferase pocket. Then go to Graphics > Labels, select your bond, click the "Graph" tab, and Click the "Graph" Button.

Question 9: Describe the data that you see, and propose a possible conclusion.

LAB 6: UMBRELLA SAMPLING

In the Introduction to Dynamics lab, you learned about performing molecular dynamics and the idea of applying constraints in the Energy Minimization lab. Here, we will use a constraint to hold two "room temperature" water molecules apart at different distances ranging from 2.2 to 5.0 Å. To keep it very simple, the environment will simply be a vacuum, but the same principles apply in condensed matter simulations.

The constraint used for this kind of assessment is called an **umbrella potential**, probably because it holds the system under the umbrella of a particular region of the reaction coordinate (or perhaps because the parabolic potential usually used is like an upside-down umbrella--we don't know for sure where the term originated). Sampling obtained under the umbrella of this constraining force is called umbrella sampling. The goal is to look for bias in the sample positions along the reaction coordinate that would reflect a slope in the underlying free energy profile.

In a system with two TIP3 water molecules, a hydrogen bond exists between the two TIP3 waters. The ideal distance between the two oxygens is about 2.8 angstroms. The interaction energy approaches zero when the two oxygens are more than 4.0 angstroms apart. We will take the O-O distance as the reaction coordinate for the formation of the hydrogen bond in the water-dimer and sample the free energy profile along this reaction coordinate.

I. Theory--Potential of Mean Constraint Force Method

The free energy profile along a reaction coordinate is equal to the reversible work of moving along that coordinate, which takes into account both the potential energies of all reactants as they move along the reaction coordinate, and also how interactions between reactants and of each reactant with the environment affects the representation (distribution) of different configurations in the relevant ensemble of configurations. Frames from a constrained molecular dynamics simulation can be taken as a reasonably representative ensemble of configurations for a point along the reaction coordinate when all of the frictional and ballistic effects of relative reactant movements are null.

The force required to hold the reactants at that point on the reaction coordinate is called the **Constraint Force**, F_c. It is equal but opposite to the force that the system is applying to the reactants. Let's think of the constraint force holding two reactants at a particular separation as if it were being applied to one atom, the test or ith atom, and think of the other as one of the atoms of the system, it being a reference point for the distance to the ith atom. In each configuration, the constraining force is:

$$F_{c,i} = -\sum_{j \neq i} F_{i,j}$$

Integration of the mean force applied by the system to the test reactant along the reaction coordinate dimension gives the work of moving along the reaction coordinate. The free energy is the energy available to do work. Equivalently, the work of moving along the reaction coordinate, from infinite separation to the reaction position, ξb, under the influence of the Mean Force is the free energy profile.

$$W(\xi_b) = -\int_{\xi=\infty}^{\xi_b} F_{c,i} d\xi$$

The type of free energy being calculated depends on the ensemble conditions used to produce the simulation trajectory. If constant pressure and temperature conditions were used, the free energy is considered a Gibbs free energy, ΔG. If constant volume and temperature conditions were used, it is considered a Helmholtz free energy, ΔA.

We will use this Potential of Mean Constraint Force method in this lab. It is important to point out that the Potential of Mean Constraint Force is NOT equivalent to the Mean Constraint Potential, which depends on the force constant of the constraint used in a non-linear fashion. The constraint energy is reported by CHARMM in the log file, but to get the mean force for integration, you must analyze your data in two steps. You will do this in an Excel spreadsheet.

First, you will calculate the average distance between reactants (water oxygens) from each umbrella and compute, knowing the force constant, $k=K_{MIN}=K_{MAX}$, the constraint force on the test atom relative to the reference atom. These will be extracted from your logfile using the unix command, grep (get the representation), which prints to standard output any line containing the string fed to grep as an argument.

Next you will use numerical integration to compute the work to move from the ξ represented by one umbrella to that represented by the next. This is not too hard: you just multiply the average of the constraint forces at the two values of ξ by the difference in ξ. The hardest part is rationalizing the sign, but you can make sure you have it right by intuition. The sum of each of the increments in work is the free energy profile. In our simulations, we will not use constant volume nor constant pressure, so it is not a Gibbs or Helmholtz free energy. The ensemble we will use is the semicanonical ensemble, and we believe that the free energy is just referred to as the free energy.

For completeness, we will also describe next a closely related approach that is often used with umbrella sampling. Used with an equivalent data set, it gives a smoother but comparable free energy profile. You can read it later. For now, skip to the instructions in Part III. Then come back and answer question 1 for your writeup.

II. Theory--Weighted Histogram Analysis Method

For a canonical ensemble the Helmholtz free energy is give by:

$$A(\xi) = -k_B T \ln \rho(\xi)$$

where $\rho(\xi)$ is the probability density along the reaction coordinate ξ.

When an umbrella potential (also called “auxiliary window potential”) $U(\xi) = k (\xi- \xi0)2$ is introduced, the free energy could be rewritten as:

$$A(\xi) = -k_B T \ln[\rho^*(\xi)] - U(\xi) - C_i ,$$

where $\rho^*(\xi)$ is the biased probability density and can be determined from the dynamic simulation trajectory. Because $U(\xi)$ is a known function, $A(\xi) + C_i$ can be calculated for each window.

Question 1: Why do we have to use the umbrella potential at all if all we need is to figure out the probability density $\rho(\xi)$?(Hint: it has to do with how long it would take to get an infinite

number of configurations for the canonical ensemble and the steepness of the underlying free energy profile).

Due to the limited computer time and power we compute a series of segments of A(ξ) that overlap each other. Each A(ξ) covers a small portion of the whole ξ range. These overlapping regions are then combined to form a continuous free energy function. This requires the determination of differences in the constants, C_i, for the windows. In practice, this has often been accomplished by shifting one free energy fragment up or down to optimize its overlap with that of the adjacent window. Alternatively, the free energy difference between ith and jth window potential $[C_i - C_j]$ can instead be computed directly from the dynamics trajectory information. Then the constant Ci for each window is calculated by the following equation:

$$C_i = [C_i - C_i\text{-}l] + \ldots + [C_3 - C_2] + [C_2 - C_1] + C_1$$

leaving only one arbitrary constant $C_{l,}$ which is usually set to zero.

III. Molecular Dynamics Stream File with Umbrella Sampling

Copy the directory for this lab to your personal directory. The skeleton of the CHARMM dynamics script is found in "ww.str_SKELETON". It is missing several key components. Your job is to fill in those components to make the file work. They are represented in the file as ########. You will copy this file to ww.str and then edit ww.str to be usable.

Let's walk through it. After opening the necessary rtf and parameter files, it sets up a loop. In each pass through the loop, a new segment consisting of a water dimer is generated. The oxygen positions are set (at a goodly distance from prior dimers) and the umbrella potential to hold the oxygens at a separation, ξ counter is applied. The NOE command block is used to apply the umbrella potential constraint. The important thing here is the ASSIGN command which applies a constraint to the two atom selections (cons.doc).

After the loop finishes, the hydrogen positions for all of the water molecules are built using HBUILD. This positions the hydrogens in their ideal locations, given the parameters for the water molecule and the forces from other molecules in the system. The spacing between dimers is sufficient that no interactions with neighbor dimers are included in the energy. Finally, the system is minimized, heated, equilibrated, and simulated. At the end, the coordinates (both CHARMM and PDB format) and PSF files are written for the post-analysis and visualization.

Question 2: What is the force constant k (the umbrella potential defined as $U(\xi) = k (\xi - \xi 0)2$) during minimization?

Question 3: If you want to do some more samples, say 5.0-5.5 angstrom with an increment of 0.1 angstrom, what changes would you make?

As we warned you, there are several #########'s in the script. These are the parts of our stream file you need to contribute. Your job is to select the two oxygens in the ASSIGN commands and finish the rest of the dynamics commands. The molecular dynamics simulation should do 0.5 picoseconds of heating, 0.5 picoseconds of equilibration and 1 picosecond of simulation. I want you to create 500 frames

in the "simul.dcd" file. This is going to be a nice test of your understanding of those awesome DYNA keywords.

Notice, we are sampling the whole reaction coordinate with equal spacing, and we are using the same constraint strength all the way through. This is OK for the purpose of this lab. But for a real project, one might have to explore different constraint strengths and sampling spacings for different regions. Of course you are very welcome to change those parameters and explore whether you can get better results.

Work on those #########s, replace them with correct codes, save the final stream file as "ww.str" and attach it with your report to the T.A. at the end of the lab.

If you are on Marylou, edit the batch.submit script to include proper input file name and then execute ww.str using **`sbatch batch.submit`**. If you are on Wolf, edit the wolfsubmit.sh script to include the appropriate file name and then execute ww.str using **`./wolfsubmit.sh`**. You can check the end of the log file to see if it's done, using the command **`tail -n 40 [log filename]`**

OK, you are now creating a DCD files with the dynamics for 57 independent water pairs, some RST files and a log file. We need the DCD files for post analysis but not the RST and log files. So remove them after you finish correcting your stream file with the **`rm *.rst`** command.

Using the rm command in conjunction with the wildcard (*) is very unforgiving. ".rst" refers to all files ending in ".rst". If you use the * by itself, it will refer to every file in your directory! When you remove a file in Linux, the file is gone!*

IV. Post-simulation Analysis

The DCD file you just created contains 500 simulation frames, each with 57 dimers at 100 Å separations. The stream file "rww.str_SKELETON" is designed to read the dcd file and output the average O-O distance (which is our ξ) for each dimer to the log file. Read the "rww.str_SKELETON" file. The only thing you need to do in order to change this skeleton into your "rww.str" is to identify the boundaries of the one loop in the file, so that you are clear on the flow of the routine, just like what you did with ww.str_SKELETON.

This routine introduces you to the CORREL facility, a CHARMM facility that is very powerful for use in MD trajectory analysis (correl.doc). It allows you to create a time series for nearly any measurable of interest. A MANTIME facility can also be invoked to manipulate the time series, for instance to get a correlation function between two measurables. Hence the name CORREL. But we will not use MANTIME here. We just want to extract the separations between water oxygen atoms for each pair. To do this, we create a time series for each pair with 500 values in each series. Maxseries tells CORREL how many of these time series it should create. For data storage purposes, we also have to tell CORREL to set aside enough values in the array for information on each of the atoms in the dimer and the code stating that it is the distance between them we want. This requires the formation of a datastructure with 3 positions for each time series, which is provided with the Maxseries keyword.

After the CORREL facility is initiated with the first CORREL command allocating memory, we next define the items we want for each time series using the ENTER command. We can do this in a loop because standard charmm commands can be executed by the CORREL facility. The first argument to

ENTER is the name of the time series for future writing or manipulation purposes. Here, we use the user-defined variable, count, to increment the time series names in the loop. Next, we specify the class of time series: distance for a series of distances between two atom selections. Finally, we select the atoms to be used, using a simplified select syntax, similar to the situation for other commands (IC SEED and CONS DIHE).

After creating the time series, we load them with data using the trajectory command, which, in the CORREL facility, reads through the previously opened dcd file and extracts all the data specified in the time series. As a side benefit, the average and RMSD for each series is reported in the log, which we can use directly for our Excel spreadsheet in a moment.

Again, go to the CHARMM dictionary or www.charmm.org and look up the new commands, CORREL, CORREL ENTER, and CORREL TRAJECTORY, and try to understand what we're doing. By now you should have developed the habit of going to the CHARMM User's Guide and CHARMM Dictionary whenever you need them.

1. When you finish replacing the ######### in your personal version of rww.str run it with CHARMM as described earlier.

2. After you get the log file, check what the final lines look like to ensure it completed without any errors. Then use the following Linux shell command to extract each of the lines containing a time series average (i.e. containing the string "Average") and store them in averages.dat:

```
grep Average [log filename] > averages.dat
```

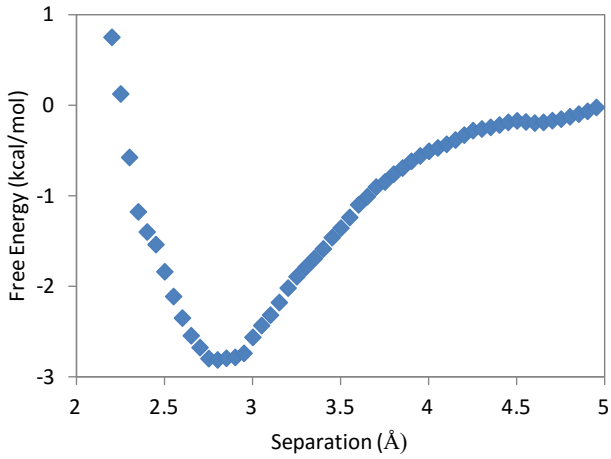
3. Now, download averages.dat to your desktop (using the download down-arrow button in the icon menu at the top of your ssh screen) and open the file with Excel, accepting the default fixed spacing between columns. Delete all the unneeded columns, keeping just the index number, the average, and the umbrella position. Now, create three new columns to the right of these three: one for the constraint force (kcal/mol-Å), one for the increment in work (kcal/mol), and one for the integral (sum) of the work increments (kcal/mol).

The first constraint force column will require that you calculate the mean constraint force from the difference between the mean separation and the desired separation using $F_c = k\Delta x$. You can create the equation for the first cell and then fill the rest of the column by clicking the dot in the lower right hand corner of the first cell and dragging down to the end of the column.

The second work increment column requires that you calculate the mean of the forces from the current and next row, and then multiply it by the change in separation for the two rows. After you have the sign right, again, fill down.

Finally, the third PMF column requires that the incremental work for each row be added to the previous row's accumulation. I prefer to do this from the bottom up, because I want my reference free energy to be that of infinite separation.

Plot the PMF (integrated work, Kcal/mol) against ξ (average separation, Å) using the Insert Chart option in Excel. Include your spreadsheet as an attachment to your writeup when you are done. Your results should be similar to the figure below.



Question 4. How do you explain the shape of the PMF?

If you have extra time, you might be interested in comparing the PMF to the adiabatic map for the water dimer. An adiabatic map is a plot of the potential energy (usually after optimization) as a function of position along the reaction coordinate. "Adiabatic" refers to the fact that there is no heat in the system: it is perfectly cold (in other words, there is no dynamic averaging of thermal fluctuations, the temperature is 0 K). The adiabatic map is usually quite similar to the dynamic average potential energy profile. The difference between these and the free energy profile must be due to entropy effects. If you wish, you can modify your code to get the adiabatic map for the water dimer (and/or the dynamic average potential energy) and plot it on the same plot as the free energy profile for comparison. If you are really ready to pass this course, you should be able to do this without assistance!

Email your answers to the questions and a copy of your ww.str and your excel spread sheet to your TA.

LAB 7: HOMOLOGY MODELING

Homology modeling, also known as comparative modeling of protein, refers to constructing an atomic-resolution model of the "target" protein from its amino acid sequence and an experimental three-dimensional structure of a related homologous protein (the "template"). Homology modeling relies on the identification of one or more known protein structures likely to resemble the structure of the query sequence, and on the production of an alignment that maps residues in the query sequence to residues in the template sequence. It has been shown that protein structures are more conserved than protein sequences amongst homologues, but sequences falling below a 20% sequence identity can have very different structure.[1]

Evolutionarily related proteins have similar sequences and naturally occurring homologous proteins have similar protein structure. It has been shown that three-dimensional protein structure is evolutionarily more conserved than would be expected on the basis of sequence conservation alone.[2]

Homology modeling has proven to be a useful tool for many research groups, because it can answer a widespread and fundamental research problem: "If a protein's primary structure is known, how can one determine the secondary, tertiary, and quaternary structure of the protein?" This is done by threading the sequence of the protein with unknown structure into the known structure of a similar protein.

First of all, where do researchers get the secondary, tertiary, and quaternary structures of proteins? The main method is x-ray crystallography. Other techniques include cryo-electron microscopy (~4 Å resolution at best), and solution and solid state NMR. A crystallographer will gather all known structural data about their protein of interest to create a PDB file that is as accurate as possible, and then publish this PDB file on the RCSB Protein Data Bank.

In this lab, you will be introduced to a simple fitting algorithm in the Swiss-PDB Viewer Deep-View program that builds the structure of your unknown based on that of the homologous residues in a PDB structure file. You may refer to the homology modeling tutorial at http://spdbv.vital-it.ch/modeling_tut.html; however, we will follow the directions below for the lab.

The **Influenza A M2 channel** is a tetrameric transmembrane protein found in viral lipid. It is the target of anti-flu agents Amantadine and Rimantadine which block infection. The wild type apo structure for the M2 channel was well defined using solid-state NMR and published in 2010 by Sharma et al with the pdb id 2L0J. 2L0J includes the inter membrane domain and amphipathic helices (residues 22-62) of the Influenza A M2 channel. Single amino acid mutations, such as S31N, can render the channel insensitive to anti-flu drugs. We will use homology modeling to create a tetrameter with a single amino acid mutation in each monomer that is similar in structure to the wild type.

I. Instructions

1. Browse to http://spdbv.vital-it.ch/disclaim.html, click Download, accept agreement, and download the proper version to your desktop.

2. Unzip the folder by clicking on it and clicking "Extract all files."

3. Navigate down one directory and click on the SPDBV executable (black icon). Hit Run if you get the security warning, and click on the About Swiss-PDB Viewer window to close it.

4. Search for 2L0J at pdb.org and download the FASTA sequences for the 2010 M2 Channel by Sharma et al. Copy the sequence to notepad and delete the first 3 primer amino acids (SNA).

5. Edit the primary sequence to resemble an S31N mutant. Note that the first amino acid in the given sequence is number 22. Exclude the B, C, and D sequences and save the new sequence as "s31n.txt."

6. Copy the lab files directory to your personal directory, and then download the four PDB files for the monomers of 2L0J from the segments folder. These monomer files are designated by a-d following the protein name.

7. Open your S31N mutated 2L0J FASTA sequence and open it in SPDBV by going to the tab "Swiss Model" and selecting "Load Raw Sequence from Amino Acids." A log box will appear, check it out and close it. It appears in the black window as a default alpha helix.

8. Open 2l0j_a.pdb in SPDBV using File, "Open PDB File…" A log box will appear, check it out and close it (ignore errors).

9. Fit your S31N mutant 2L0J FASTA sequence 1 to 2l0j_a.pdb in SPDBV using Fit, "Magic Fit." The alpha helix is turned into a replica overlying 2l0j_a. Additional controls of the view and the fit can be introduced using the Control Panel, which is opened using Wind, "Control Panel." The help "?" icons for both the Control Panel and the main toolbar give introductory instructions and directions to the User's Guide.

10. Save your new structure to your desktop using File, "Save" -> "Current Layer" as s31n_a.pdb. Under File, hit Close twice to clear both molecules.

11. Repeat steps 6-10 for the second, third, and fourth sequences of 2L0J fitting them to 2l0j_b, 2l0j_c, and 2l0j_d.pdb respectively.

12. In the lab directory you will find several directories for organization purposes and scripts written for your use. Import your S31N mutant files into the "segments" directory. In the "str" directory open "builder.str" and edit this script to read in your S31N mutant PDB files. This script creates a coordinate file, principal structure file, and PDB file of the final S31N tetramer from your four S31N monomer PDB files. Look through this script and familiarize yourself with any CHARMM commands or syntax that are new to you.

13. Open and revise wolfsubmit.sh or batch.submit and then execute "builder.str" using correct syntax. If you are using Marylou, you may use the `./[filename]` in place of `sbatch [filename]` for short jobs such as these to run the script on the internode rather than submit it to the scheduler.

14. Check the "channels" directory for the output files. Did your script run successfully? If not, look for the log file in the "log" directory to begin troubleshooting what might have gone wrong.

15. Now perform steps 12-14 for 2L0J, the wild-type channel.

16. In the "str" directory you will find one "dyna" script that will perform minimization, heating, and equilibration for your 2L0J wild-type tetramer. You may be required to change one of the variables for the script to run successfully. Submit the script and view the log file to ensure it ran properly. You will want to refer to the log file to answer questions later in the lab.

17. Download 2l0j.psf (in channels directory) and 2l0j_2eq.dcd (in dynafiles directory) to your computer.

18. Open VMD, load the .psf file, and then add the .dcd file to the psf. Click the Extensions tab, navigate to Analysis, open RMSD Visualizer Tool. Check "Frames from..." box, check "Backbone" box, click RMSD, then (once RMSD finishes calculating) click Align.

19. Click "Heatmap plot." Take a minute to understand the plot, and zoom out if necessary. Change the max threshold to 6, hit "apply," and save the heatmap plot.

While you can render from VMD, it is easiest and sometimes better to take a screenshot. The heatmap plot can be saved as a .gif with the File menu, but it loses the axes.

20. Repeat steps 16-19 for S31N channel.

Question 1: What is the total energy of the wild-type tetramer before minimization? How about the mutant? Why are the energies so different? What are the energies of each species following minimization? Is this energy potential or kinetic energy?

Question 2: Watch both trajectories for wild-type and mutant M2 in VMD, and comparWhy might an S31N mutant change drug binding affinities? Hint: M2 blockers bind to the inner pocket of the M2 channel.

Question 3: What are the main differences in the heat plots? Why might a simple substitution mutation give rise to these differences?

References

1. Chothia, C; Lesk, AM (1986). "The relation between the divergence of sequence and structure in proteins". EMBO J 5 (4): 823–6. PMC 1166865. PMID 3709526. //www.ncbi.nlm.nih.gov/pmc/articles/PMC1166865/.

2. Kaczanowski, S; Zielenkiewicz, P (2010). "Why similar protein sequences encode similar three-dimensional structures?". Theoretical Chemistry Accounts 125: 543–50.

3. Sharma M, Yi M, Dong H, Qin H, Peterson E, Busath DD, Zhou HX, Cross TA. Insight into the mechanism of the influenza a proton channel from a structure in a lipid bilayer. Science. 2010;330:509–512.

LAB 8: FREE-ENERGY PERTURBATION

In this lab, you will integrate the skills you have developed and the knowledge you have gained so far this semester to study the M2 proton channel of the Influenza A virus. Upon completion of this lab, you should be able to accomplish the following objectives:

- Understand the TSM perturbation syntax and methodology
- Perform TSM simulations of amantadine and rimantadine in water
- Compare ΔG from water TSM simulations to given ΔG's from channel TSM simulations to obtain ΔΔG's for amantadine and rimantidine.
- Analyze TSM trajectories in VMD

In order to complete this lab, you must use Marylou, the BYU supercomputer.

I. Background

The M2 proton channel of the influenza viral membrane is a key component involved in viral replication. When the channel is blocked by channel inhibitors, such as Amantadine or Rimantadine, the virus cannot replicate. In mutant forms of the virus such as S31N, these drugs are no longer effective due to amino-acid changes in the M2 channel. Understanding the M2 channel is, therefore, essential in searching for new, effective anti-viral drugs.

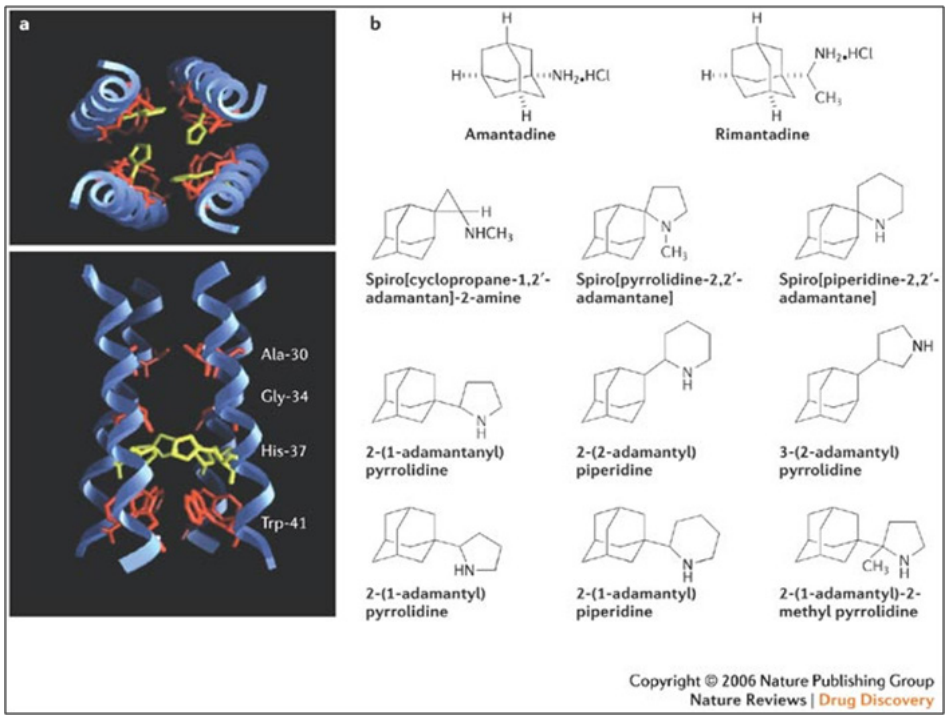


Figure 1: a) Superior and lateral views of the M2 proton channel.
b) Various adamantyl derivatives.

Free-energy Perturbation

In the umbrella sampling lab you were introduced to the idea of determining free energy along a reaction coordinate. This lab, however, focuses on determining free energy in a system using **free-energy perturbation.**

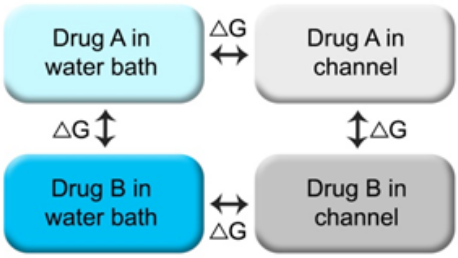


Figure 2: Determining ΔG

In practice, ΔG is usually calculated by comparing a difference in energy for a molecule of interest as conditions in the system change. However, in free-energy perturbation, ΔG is determined by keeping the system conditions constant and changing the molecule of interest into another molecule via piecewise mutations. For example, one might determine the ΔG for Drug A by comparing Drug A in water and Drug A in the M2 channel. The same effect can be accomplished by comparing Drug A in water to Drug B in water. (See Fig. 2.)

TSM

CHARMM has two methods of performing free-energy perturbation, the first is using the command PERT, and the second is using the command TSM (perturb.doc, pdetail.doc). We will use **TSM** (thermodynamic simulation method) free-energy perturbation in this lab to compare the relative ΔΔG's of amantadine (labeled ALM-035) and rimantadine (labeled ALM-150).

TSM operates using degrees of mutation, termed lambda, for which a percentage of Drug A is compared to a percentage of Drug B. The values of lambda used in this lab are 0.05, 0.125, 0.500, 0.875, and 0.95. Using thermodynamic perturbation, midpoints between the lambdas, as well as endpoints 0 and 1, can be predicted, giving ΔG at the following points:

- λ = 0.0 (100% Drug A)
- **λ = 0.05**
- λ = 0.0875
- **λ = 0.125**
- λ = 0.3125
- **λ = 0.5**
- λ = 0.6875
- **λ = 0.875**
- λ = 0.9125
- **λ = 0.95**
- λ = 1.0 (100% Drug B)

II. TSM for Drugs in M2

Before we can use the free-energy perturbation method, the systems we need to simulate must first be created. Fortunately, the systems you will be simulating are provided in the lab. Once you have obtained good systems for simulation, you will (1) heat, (2) equilibrate and perform TSM, and (3) execute a post-processing script to interpret TSM data.

For the purposes of this lab, you will not perform any simulations of the drugs in the M2 channel. Rather, you will simulate the drugs in water and compare your results to M2 results that are given to you in the charmmlab/lab_files/lab_files/tsm_output directory.

Open the tsm_output directory described above and examine the contents (do not edit any of the files here, and do not copy the directory to your personal directory). The output directory contains the output .crd, .dcd., restart, tsm, and .xtl files from the TSM scripts for the drugs in the M2 channel (2kqt).

The lab directory you will be working with (labeled "perturbation") can be found in the charmmlab directory where the other labs are stored. In this directory you will find CHARMM scripts (prefix=tsm), a customized submission script (submit.sh), and four directories. The output directory here corresponds to the tsm_output directory and will be filled by data from your personal submissions. The alm folder contains the .pdb's, .prm's, and .rtf's of the drugs you will be simulating, as well as stream files defining cage atoms. The prep directory contains coordinate files and others essential to the lab, but these will not require editing. Finally, the log directory will contain log files from your CHARMM scripts. Copy the perturbation lab directory to your personal directory.

This lab utilizes a multi-purpose submission script tailored for the lab. As you perform research, there will be many occassions where you will need to submit up to thousands of analogous simulations with similar, but varying parameters. Invoking a Linux shell script can help to submit these scripts in bulk. Open the file "submit.sh" and look over the contents. You don't need to know what every Linux command means, but acquaint yourself with the variables, for you will need to adjust them later. When **`export`** is used, a variable is passed for use by another script--in this case to a procedurally generated file via a Linux here document, which submits commands and parameters to a compute node on the supercomputer. The loops and if tests that follow cause the submission to be performed multiple times depending on the values set in the variables section. The here-doc section contains a line **`#SBATCH`**, without which the file could not be submitted to the scheduler (not to be mistaken for a Linux comment). It loads the required environments using **`module load charmm`** and finally invokes CHARMM following the general pattern below:

```
mpirun $(which charmm) [variables to pass] < [input] > [output]
```

- **`mpirun`** invokes the default Message Passing Interface (MPI) parallel environment
- **`$(which charmm)`** is a variable stored that points to the CHARMM executable in the supercomputer apps directory, set when loading the CHARMM module. When MPI is not invoked with mpirun, CHARMM will run in serial.
- **`[variables to pass]`** is an optional list of variables to send to CHARMM in the format variable:value.
- **`[input]`** is the name of the file to send to CHARMM, usually a .str or .inp file.
- **`[output]`** is the name of the file to write CHARMM output to, usually called a log file.

Heating

Open the file labeled "tsm.heat_2kqt.str." This file is the CHARMM script used to heat the drugs in the 2KQT model of the M2 proton channel in a lipid bilayer. You should be familiar with many of the commands seen in this script from previous labs. You will also notice the use of **absolute references** and pathnames, rather than the usual **relative references**, to allow CHARMM to access the files from any directory on the supercomputer. Variables not defined in the CHARMM script are passed from submit.sh, as described previously. Use what you've learned in the previous labs to understand what each command is doing in this script, and consult the CHARMM documentation or the TA's if necessary.

Question 1: What are the dimensions of the crystal for this protein system?

Question 2: You will notice a couple of lines that use the CHARMM command "define." What do you think this command does? Describe what it influences later in the script.

The file labeled "tsm.restart_2kqt.str" takes restart information from the previous script and continues the simulation using CPT dynamics for equilibration.

Equilibration & TSM

Now the system has been heated and has begun to equilibrate, the equilibration process can be extended to allow for calculation of ΔG. Equilibration is performed in a step-wise fashion using **restart files**. You can imagine the way restart files are used just like working on a lengthy project in a word processor. It is ideal to save often and pick up your work where you previously saved, rather than completing it all at once without saving at all during the process.

> *Question 3: In theory, you could use one script and one submission and extend the equilibrium dynamics process to a longer amount of time. However, this is often an impractical solution. Name two reasons that you can think of that make the use of restart files advantageous.*

The file labeled "tsm.restart_2kqt.str" in your lab directory is the script that would be used to extend equilibration time for the system and perform TSM calculations. Restart scripts can be made to take more than a day to complete on the supercomputer--for this reason you will not be required to edit or submit this script. The coordinate files, dcd files, tsm files, and more from the 2KQT simulations are found in the tsm_output directory in the lab_files directory from earlier. Now that you've familiarized yourself with the restart and heating scripts, answer the following question:

> *Question 4: What variable is passed to the CHARMM script from the shell to allow the restart script to load the output files from the heating script? What would the value of the variable need to be for the second restart after the first post-heating restart?*

Post-Processing

TSM's output files in charmmlab/lab_files/tsm_output/tsm/ are not in a format readily understood for analysis. For this reason, TSM has its own CHARMM post-processing method that it employs to give a printout of ΔG. You will find in your lab directory a file named "tsm.post_processing.str." This script loads up TSM output files with a loop, calculates midpoints based on given lambdas, processes the TSM output files with `TSM POST PSTAck 10 PLOT`. The submission script, "submit.sh," sorts the TSM data using Linux commands depending on the variables the user sets at the start of the file. The postp folder contains the organized output data for analysis with programs such as Excel. Delta A (which in this case really means ΔG) is the free-energy data you are interested in, Delta E is the change in energy, and Delta S is the change in entropy.

III. TSM for Drugs in Water

Now that you have familiarized yourself with how to heat, equilibrate, and analyze TSM simulations for drugs in the M2 channel, ***perform these same steps for drugs in water***. This is a necessary in order to determine the free-energy of binding, ΔΔG, given by the difference in ΔG between the channel and water. The submit.sh script needs only small adjustments, the batch.submit and tsm.post_processing files need no adjustment, and two CHARMM input files for water (with the same naming pattern) will need to be made. You may copy a lot of the code from the 2KQT heat and restart scripts (by copy-pasting or using the cp command), but you will need to make adjustments to ensure they work properly for the drug-water systems.

Notice the minimization methodology for M2 will not apply to drugs in water, so create your own minimization procedure (don't overthink this). You only need to perform one restart, but you may perform more than one if you desire. Depending on the usage of the supercomputing clusters available on the marylou.byu.edu home page, you may desire to adjust the sbatch command at the end of the submit.sh script to improve your odds at entering the queue and getting compute resources or to accelerate the simulations (see Heating in Lab 5 for description). Use all of your available resources, scripts, and output to figure out how to do this with as little help from the TA's as possible

IV. Analysis

The data analysis for this lab takes place in two parts. You will begin with a graphical analysis of data and then finish with a visual analysis of the system in equilibrium.

Statistical Analysis

Import the post-processing output files for the drug in protein and the drug in water (found in tsm_files) into the statistical program of your choice (Excel, Matlab, etc.) and compute the ΔΔG's (protein minus water) for amantadine and rimantadine (refer to the background section).

Question 5: Which drug has a lower ΔΔG? By how much?

Prepare a scatterplot showing ΔΔG per value of lambda and include it in your lab writeup.

Visual Analysis

Open charmmlab/lab_files/tsm_output/crd/ and select the coordinate file for the 2kqt system in equilibrium. Load this file in VMD (be sure to specify CHARMM coordinates) and add the corresponding DCD file to the molecule. Press play and inspect the molecule and its trajectory. Use labels and the representations "SEGID M2A M2B M2C M2D" and "RESNAME L035 L150" to help you find the channel and the drugs. Answer the following questions:

Question 6: Toward which terminus (N or C) does the drug orient in M2 while in equilibrium?

Question 7: What amino acids do you find nearest to the drug while the system is in equilibrium?

Feel free to explore the molecule more and see what other interactions may be occurring.

Question 8: Name two things you could change in the scripts or in your methods to collect more accurate and reliable TSM data.

GLOSSARY

.DCD FILES (6)
File used to view an MD simulation or other trajectory generated by CHARMM.

.PDB FILE (6)
Protein Databank File that contains coordinates and bond information of a molecule or system.

ABSOLUTE REFERENCES (58)
File or directory path that originates at the root directory of the Linux system.

ADOPTED-BASIS NEWTON-RAPHSON (ABNR) (12)
A derivative method of minimization best suited for large systems.

CHARMM (2)
A molecular simulation program developed by Chemistry at HARvard Macromolecular Mechanics.

CLIPPING PLANE TOOL (5)
A VMD tool used to view cross-sections of molecules of interest.

COMPARISON SET (12)
A coordinate set used to compare differences, apply constraints, and more to a system of interest.

CONDITIONAL STATEMENTS (16)
Used to execute portions of a script based on user-defined criteria.

CONJUGATE GRADIENT (CONJ) (37)
A minimization procedure somewhat more optimized and calculation-intensive than SD.

CONSTRAINT FORCE (46)
The force required to keep a reactant at a specific point in a reaction coordinate.

CONSTRAINTS (38)
Energy terms that apply a restoring force, allowing one to confine a selection to certain coordinates.

ENERGY MINIMIZATION (12)
A set of procedures that changes atom coordinates to find local minima.

FATAL ERROR (21)
A CHARMM error that forces the stream file to exit prematurely.

FORTRAN LOGICAL UNIT NUMBER (12)
A temporary read/write space used by CHARMM when executing stream files.

FREE-ENERGY PERTURBATION (57)
A method for measuring the F.E. difference (often ΔG) as a molecule changes one state to another.

HOMOLOGY MODELING (52)
A modeling method used to construct a protein from a sequence differing slightly from a template.

INFLUENZA A M2 CHANNEL (52)
A tetrameric, proton-selective channel protein that is essential to viral replication.

INSERT MODE (8)
A mode used in Linux' Vi command to add, remove, and edit text. Accessed by pressing [i] in Vi.

LOG FILE (13)
A CHARMM log file shows the results of the stream file execution and is essential for troubleshooting.

LOOP (16)
A loop is a portion of a script which repeats according to user-specified conditions.

MOLECULAR DYNAMICS (40)
Simulations that contain a dynamic component which varies over time; e.g. heat, pressure, etc.

ORTHOGRAPHIC MODE (3)
A VMD display mode which removes persective and portrays objects using constant dimensions.

PATCH (34)
A CHARMM feature that allows for the PSF of a segment to be manipulated.

PDB ID (2)
A unique identification code given to every molecular structure on the RCSB Protein Databank.

PERIODIC BOUNDARIES (7)
Takes a region of a system and creates copies of itself around it. Most useful for molecules in solvent.

PERSPECTIVE MODE (3)
A VMD display mode which portrays objects with dimensions that vary with distance from the viewer.

PRINCIPLE STRUCTURE FILE (PSF) (10)
Contains important bonding information for the structure of segments in CHARMM.

RCSB PROTEIN DATABANK (2)
A commonly used online repository of 3D structures such as proteins, nucleic acids, etc.

RELATIVE REFERENCES (58)
Linux paths that originate in the directory of command execution.

RENDER (5)
A VMD command which creates a bitmap snapshot of the OpenGL scene window.

REPRESENTATION (4)
User-defined display properties given to a selection of atoms in VMD.

RESIDUE TOPOLOGY FILE (RTF) (23)
A file which defines the type, mass, and charge of atoms in each reside, allowing for PSF generation.

RESTART FILES (59)
MD output files that contain information for continuing a trajectory.

RMSD (44)
Root-mean-square deviation, used to determine the difference between structures or conformations.

SELECTION (15)
A CHARMM procedure for applying calculations, changes, etc. to specific atoms in a system.

SLURM (13)
A Linux scheduling system used for submitting multiple jobs to CHARMM on Marylou and Wolf.

STEEPEST DECENT (SD) (36)
A minimization procedure that locates the nearest local energy minimum for a structure.

STEREO VISION (4)
A VMD display mode which overlaps two images of an object, creating an illusion of a 3D shape.

STREAM FILE (9)
An input file (usually with the extension .str) which, when executed, submits input to CHARMM.

TSM (57)
A CHARMM protocol that facilitates the conversion of one molecule to another in FEP.

UMBRELLA POTENTIAL (46)
A constraint used in umbrella sampling procedures for determining binding energy between molecules.

VARIABLE (15)
A character string which is assigned a user-determined value.

VMD (2)
3D Molecular visualization software used for analyzing biomolecular systems.

WILDCARD (15)
A character with specific interpretations in the Unix shell. Includes characters such as ?, *, ~, etc.

WOLF (8)
Dr. Busath's Linux cluster used for the CHARMM lab.

LINUX CHEATSHEET

- man [command]: Gives manual information on a command
- ls: Lists the contents of current directory. Use -l to give more details.
- cd [directory name]: Allows changing of directories. Use ../ to go back a directory.
- mkdir [directory name]: Creates a directory.
- rmdir [directory name]: Removes a directory.
- rm [filename]: Removes a file.
- cp [original filename] [new filename]: Copies a file.
- mv [original filename] [new filename]: Moves a file.
- vi [filename]: Opens a file for viewing or editing.
- grep [pattern] [file]: Finds an expression in a file.
- *: Match any string of characters.
- ?: Match any single character
- pwd: Gives directory tree for present working directory.
- chmod 777 [filename]: Changes permissions to read, write, and execute for all users. Useful when a file says “permission denied.”
- sed ‘s/[originalstring]/[newstring]’ [filename]: Replaces [originalstring] with [newstring] anywhere in a file.
- squeue: Shows all the jobs in the SLURM queue. Use “watch squeue” to update this list every two seconds.