**BIOS 10603 Lab 8**

**Analysis of Your Protein’s Production Simulations**

Name:

Please insert all answers and figures into this Word document, and then save the completed lab as “Lastname\_Lab8.docx”. When you have completed the lab, submit it as an attachment to the Lab-8 assignment link on Canvas. The report is due by midnight the day after your lab period.

**Objective**

Over the past few lab sessions, you have submitted both your protein (control) and a secondary system for production simulations (NPT). In this laboratory, we will concatenate (or “glue”) the output trajectories together and analyze the entirety of your simulations thus far.**As in previous weeks, throughout this lab, we will use “XXXX” to refer to your protein’s PDB code***.*

**Lab 8 Sections:**

**1.** Concatenating the trajectories

**2.** Unwrapping your trajectories

**3.** Trajectory Analysis

**4.** More VMD script analysis

At this point in time, you should have accumulated 40 ns of simulation time for both systems. If you have not completed your production simulations, you should do this lab with the **last NPT simulations that have been completed successfully.** However, you will need to re-run this lab analysis after you have finished the full 40 ns of production time for your final project.

**Part 1: Concatenating the trajectories**

**Creating a protein atom index file:**

The first task today is to concatenate all of your trajectories, **beginning with the heat up/equilibration and continuing through all of the NPT simulations,** together into one large trajectory. We will do this using a program called catdcd after creating a custom index file for your protein that will remove any water molecules and ions from the simulation dcd files. If you think that your protein has some special interaction with water (such as anti-freeze proteins) or if you included ligands, please talk to your TA or Dr. Haddadian before generating this index file.

First, copy the file called catdcd and index\_generator.py from Canvas to your home directory on Midway. After you have done this, find the XXXX-autopsf.pdb file for your protein (the PDB file you created before solvating step). There are two way you can count of number of protein atoms.

1. Open this file in a text editor and scroll to the end of your protein’s atoms: we are interested in finding out how many protein atoms are in your system.
2. Type the following two lines of VMD code in the TkConsole:

set sel [atomselect top protein]

$sel num

**Q1.** Try both methods listed above.

1. Are there water molecules in your initial PDB file? How can you know that? If your file contains water molecules, let either your TA or Prof. Haddadian know
2. How many protein atoms are in your system? Do the two values you obtain match? If not, let your TA know immediately.

After you have found the number of protein atoms in your system, open up the index\_generator.py, and replace the numbers marked “1111” with the number of atoms in your protein, and then save the file. If your protein has, for example, 5242 atoms, then the line should be changed from:  
for i in range(1111):  
*to*

for i in range(5242):

When this is completed, execute the following commands in the Terminal:  
module load python

python index\_generator.py > index.txt  
  
This will create the file index.txt in your working directory, which simply has the atom indices of your protein atoms, which will be useful to concatenate the trajectories together and remove the water molecules and ions. We will use this as the index file. **Repeat the same procedure for your second trajectory.**

Our next step is to locate all of the dcd files from your trajectories to date (you will probably have a total of 3 trajectories for each system). To concatenate trajectories together, we use the catdcd program. The syntax for calling catdcd from the command line is as follows (all on one line):

./catdcd -o outputname.dcd -i index.txt -stride 10 inputfile1.dcd inputfile2.dcd … inputfileX.dcd

Inputfile refers to the prefixes of your dcd files. To concatenate your simulations, we simply need to replace the appropriate values from your trajectories and then issue the command. Specifically, you need to change:  
**A.** The stride – this determines how often a new frame is written (choose a value of 10)  
**B.** The outputname.dcd – choose something informative. We suggest XXXX-heatup-equil-NPTYtoZ-stride10.dcd  
**C.** The inputfile.dcd – These should be **in the right order order as: heatup-equilibration, and then NPT simulations**. For example, XXXX-mini-heatup-equil.dcd XXXX-mini-equil-npt1.dcd XXXX-mini-equil-npt2.dcd ....

The … indicates that the command will accept a variable number of dcd files.

**Q2**

1. What does the python script do? xplain your answer.
2. Why do we produce the index file by only counting the number of protein atoms?

If everything has gone well, your trajectories should be concatenated within a few minutes and some summary statistics will be printed out.

**Q3.**

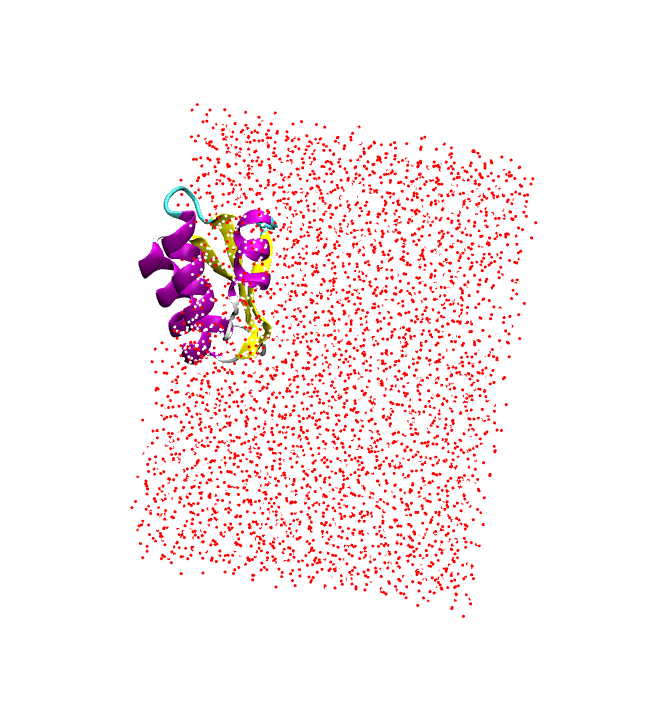
1. How many frames are in your combined trajectory?
2. How many ns of simulation time does each frame correspond to? Show your calculations.
3. What is the total size of your combined trajectory? (Hint: The ls –lrth command will print file sizes to the terminal window, in the 5th column).

Picture 1

1. Why is the total size MUCH smaller than the sum of your NPT dcd file sizes?

**Repeat the concatenation for your second trajectory**; download your combined trajectories and load them into VMD. Please note that since we have removed all water molecules and ions from the trajectory, you need to use your **initial protein PSF file for the protein** (from **before solvating** the system) to load your trajectory.

**Part 2:** **Unwrapping your trajectories**

****If, at any point in time, your protein appears to leave the water box (see right) and then suddenly jumps to the other side of the water box, **you need to unwrap the protein**. In the simulations, we set boundary wrapping conditions, so if a molecule moves too far and exits the box, then the molecule will enter from the other side. This allows for keeping the shape of the water box as a cube. This introduces artificial artifacts to the final analysis of the system, so we need to unwrap the trajectory. To do this, run the following command in the TK Console on your concatenated DCD file after you loaded it into VMD:

pbc unwrap -first 0

This command unwraps the protein with respect to the first frame and may take a few minutes to complete. If your protein has some special interaction with water (such as anti-freeze proteins), please talk to your TA or Dr. Haddadian before unwrapping your trajectories.

After the unwrapping is done, align your concatenated trajectories to the last frame of their corresponding minimization (one at a time) and save the trajectory for further analysis. Remember, you should use the RMSD trajectory tool to do the alignment and use the “Save Coordinates” button to save your trajectory. **Make sure the file type is DCD. You can name the new file as**

XXXX-heatup-equil-NPTYtoZ-stride10-align.dcd.

**These two DCD files are the main files that we will use for analysis as described below.**

**Part 3: Trajectory Analysis**

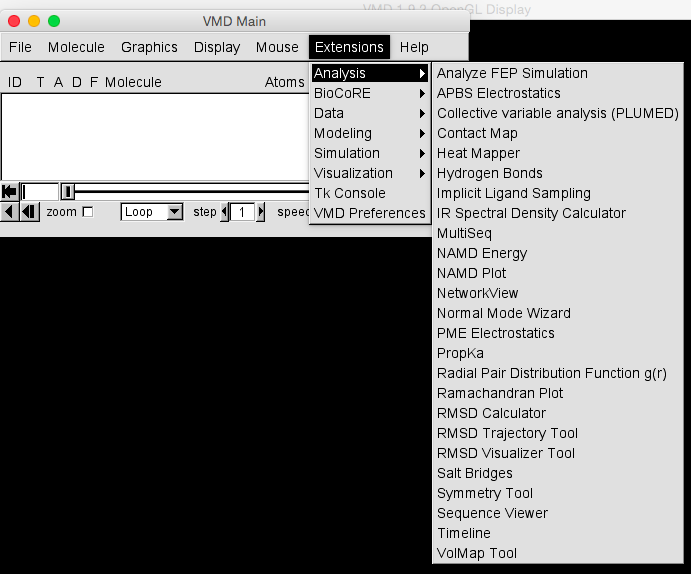
First, you should visualize the trajectory in VMD. VMD assigns the secondary structure (-sheets and -helices) to the first frame and keeps the same assignment through the trajectory (to save memory space) even as the secondary structures change. This is misleading and needs to be corrected. Download the sscache.tcl script from Canvas and source it in the VMD TK console (this script keeps the secondary structure assignment for all frames). Then run the script with the following command

start\_sscache molid (where the molid is the number assigned to the protein by VMD)

Then play your trajectory to allow VMD to recalculate secondary structures. **If this takes a long time, you can turn off the display in the main window by double-clicking on the “D” next to your loaded molecule.**

It would be best to show the protein with “new cartoon” and color it with “secondary structure” before you check the trajectory. **View the trajectory a couple of times (with different speeds), zooming in on the important parts of your protein.**

Note that in this lab, since we are producing data that will be used in your final project report, it is important to have the proper axis labels. **In particular, the frame number is a rather meaningless label on the x-axis and should be replaced with the actual simulation time (nanoseconds)**. You should multiply the frame number by the appropriate number of nanoseconds per frame (**and in case you concatenated the trajectory with stride, don’t forget to take that into account**), as determined in Question 2, and use that for your x-axis. For example, in Excel, you can use a formula entry to multiply the value in the cell A2 by 50 (A2×50), and then you can drag the bottom right corner downwards to auto-fill this formula. In R, you can issue the command “infile[,1]= infile[,1] \* 50” to multiply the first column of a table by 50.



VMD has a host of different analysis tools that you can use; feel free to explore these for your final report. Also, see the link below

<http://www.ks.uiuc.edu/Research/vmd/script_library/>

In all of the analyses below, you need to use your concatenated DCD files. And you should include pictures of all of them.

**All of the analysis tools below can be focused on the important parts of your protein (such as the active site, mutation site, etc.) by defining the residue range. This is helpful for your final report.**

In this lab, we are interested in the following analyses, which you have already performed in previous labs:

**Q4.** **RMSD** – Using R, plot the RMSD for your control and the secondary system **on one graph**. Compare and contrast the two plots and explain any trends you observe and try to correlate them to a visual analysis of the trajectories.

**Q5.** **RMSF –** Calculate andplot the RMSF **for the stable regions of your trajectory** (as determined by your RMSD plot as you did in lab-6) **on one graph**. Compare and contrast the two plots and explain any trends you observe and try to correlate them to a visual analysis of the trajectories. **Remember that before any RMSF calculation, you need to align the trajectory to the first frame of the stable part of its simulation.** Remember, the stable part of a trajectory is the frame range where the RMSD values are more or less constant (after the initial large jump).

**Q6.** Coloring the residues by their RMSF values is a very good way to visualize and pinpoint the important mobile parts of the protein. Download RMSF-color.tcl from Canvas and make the appropriate replacements. This script adds the residue RMSF values to the “Beta” column of a PDB file. Paste the 5 lines below. Please don’t paste in the entire file, just the 5 lines you changed/added.

**Q7.** Now, we want to color each residue by its RMSF values to better visualize the flexible regions of the protein. In the above script, we set the value of the Beta column for each residue to be equal to its RMSF and saved a new PDB file. Load this file into VMD, and in the “Graphical Representations” window, set the drawing method as “NewCartoon” and the coloring method as “Beta.” Then go to “graphics”🡪“colors” 🡪 “color scale,” and choose “BWR.” Blue indicates areas of your protein that have low values of RMSF, while red indicates areas of your protein that have high values of RMSF. Repeat the same procedure for your secondary trajectory and take a screenshot of your colored proteins in both systems and paste them below. Does this coloring agree with the plot that you produced in Question 5? **For your report, you need to focus on the regions with large RMSF values (peaks) in the middle of the protein for both systems and compare and contrast these regions.**

**Q8.** **Rg –** Plot the radius of gyration for your control and secondary system **on one graph**. Compare and contrast the two plots and explain any trends you observe and try to correlate them to a visual analysis of the trajectories.

Salt bridges can be divided into strong and weak ones. A strong salt bridge is one that is stable pretty much throughout the trajectory. In contrast, a weak salt bridge is not stable (it forms for a short time and then breaks).

**Q9.** Use Timeline to analyze the salt bridges formed in your protein for both of your trajectories.

1. Do the salt bridges shown in Timeline correspond with any important regions in your protein (the information can be found in the paper reference of your protein)?
2. List one stable (strong) salt bridge and describe how it changes during the simulation. Include a screenshot below.
3. Are there any differences between the number and trend of salt bridges between the two systems?

There is an alternate way to analyze the salt bridges in your protein. Go into Extensions 🡪 Analysis 🡪 Salt Bridges. In the window that appears, make sure your protein is selected and the box stating “Write a file with the distances for each salt bridge” is checked. Click “Find salt bridges.”

**Q10.** Pick one of the stable (strong) salt bridges that were output by the program in your control system and make a plot of the salt bridge distance vs. time. Does the same salt bridge exist in your secondary simulation? If yes, add the distance vs. time graph of this salt bridge to the first plot and compare and contrast the two graphs; is the salt bridge stable in your secondary simulation? If not, explore why the salt bridge does not exist in the second simulation. (Hint: a strong salt bridge is one that maintains a relatively constant distance (~3.2 Å) between the atoms involved through the course of the simulation). Include your graph below.

**Q11.** Use your own computer or Timeline to analyze the secondary structure of your protein in your control simulation.

1. How does the secondary structure of your protein change over time?
2. What are the important regions of your protein, and how do these regions change?
3. Are there any regions in the protein with changed secondary structure over the course of the simulation? Include a screenshot below.

**Q12.** Repeat Q11 for your protein in your second simulation.

1. How does the secondary structure of your protein change over time?
2. What are the important regions of your protein, and how do these regions change?
3. Are there any regions in the protein with changed secondary structure over the course of the simulation? Include a screenshot below.

**Q13.** Compare and contrast the secondary structure between the two systems and describe the regions of difference. Are these regions near the important parts of your protein? Explain the trend.

**Q14.** **Hydrogen Bonds (“H-Bonds”):** Using the H-Bonds tool built into VMD (Extensions 🡪 Analysis 🡪 Hydrogen Bonds), find the number of hydrogen bonds in your protein backbone over the course of the simulation for both of your systems. To achieve this, choose atoms “name N O” in the selection box and use a Donor-Acceptor distance of 3.5Å and an Angle Cutoff of 35 degrees. Plot the number of **H-Bonds** for your control and the secondary system **on one graph**. Compare and contrast the two plots and explain any trends you observe and try to correlate them to a visual analysis of the trajectories.

You may find it helpful to create a visual representation of the hydrogen bonds in VMD. First, make a “new cartoon” representation of your protein and then add another representation. In this new representation, choose “name CA C N O HN” and show it with licorice. Finally, add another representation with “HBonds” as the drawing method and change the distance and angle cutoff as above (3.5 and 35), and then change the line thickness to 5 and the color to yellow.

VMD has a nice feature called visualization state that allows you to save the changes you made to the protein (like coloring) for future usage. Saving the visualization state means that when you load it, you will resume working with the exact representation that you saved. This means the coloring will be the same, the viewpoint will be the same, and all of your graphical representations will be saved. To save a visualization state, go to File 🡪 Save Visualization State, and choose a filename. The file extension will be “.vmd” To load this visualization state, choose the file from File 🡪 Load Visualization state.

**Part 4: More VMD script analysis**

As you saw before, VMD has an extensive [script library](http://www.ks.uiuc.edu/Research/vmd/script_library/). This library contains a lot of user-submitted scripts to do various functions in VMD that you may find it useful for your final project. One of the useful scripts is [fit\_angle.tcl](https://www.ks.uiuc.edu/Research/vmd/mailing_list/vmd-l/att-2279/fit_angle.tcl) which calculates the angle between the best-fit line through two selections or between a selection and a given vector. Delete all of your trajectories and load in your Ubiquitin heat-up/equilibration simulation into VMD.

**Q15.** Open the script in a text editor and edit sel\_angle\_frames such that it will put the angle for each frame into an output file with the corresponding frame separated by a tab. Paste your code below:

**Q16.** Calculate the angle between the best-fit line through the ubiquitin large helix and the z-axis for all frames. Plot the results in R and **comment on any trends you see**.