## **1: Converting MD Trajectories into Dihedral Trajectories**

**Note**: Commands will be described so you know what’s going on under the hood. Scripts that run the important commands for you are provided in the repository.

### **1.1: Trimming Raw Trajectories**

**1.1.0: Checking Trajectories**

Before and after the production run is done, be mindful and check your simulation results. You should use “gmx dump” to check your inputs. You also need to load your trajectories into VMD and visualize them to make sure nothing strange has happened such as a deformed water box with a cavity, or abnormal peptide structures.

It is important to check that your simulations did not contain any frames where your peptide has a conformation with *cis* peptide bonds, and also that the chiralities of the amino acids in the simulated peptide sequence match what is expected. To check your trajectories, both biased and neutral, for these issues, use the pair of scripts check\_trajComment.py and check\_chiralComment.py in the repository. When using these scripts, be sure to understand what use cases they are designed to handle – as discussed in the README files – and what software dependencies they have; be sure to run them with Python 3. The scripts are designed to handle protein-only systems, so you should trim your trajectory using commands like discussed in Section **1.1.1**, reproduced here for your convenience:

gmx\_mpi trjconv -f s1\_prodX.xtc -s s1\_topolX.tpr -o s1\_trimmed\_prodX.xtc -pbc mol,  
and select the atom group corresponding to your protein only system when prompted by GROMACS.

When providing a .gro file, remove any solvent water molecules (and accordingly update the value in the .gro file indicating the number of atoms). The specific atom positions are not important for these scripts, only the atom names and numbers.

The python scripts use GROMACS commands within them to process your trajectories, as well as Python 3. Index files which list the frame numbers of your trajectory with and without problems will be created when running the scripts. If your trajectory was called prod12.trj, these index files would be prod12\_cis.ndx and prod12\_trans.ndx for check\_trajComment.py, and prod12\_bad\_chirality.ndx and prod12\_good\_chirality.ndx for check\_chiralComment.py. Example usage:

python check\_trajComment.py –gro prodProteinOnly.gro –trj s1\_trimmed\_prodX.xtc –cyclic True –cutoff 120

python check\_chiralComment.py –gro prodProteinOnly.gro –trj s1\_trimmed\_prodX.xtc –seq GNSRV

**1.1.1: Creating Trimmed Trajectories**

Now that your production run has been completed successfully, you can analyze your simulation trajectories in more depth. To begin the process, you will first need to make modifications to the trajectories of your neutral replicas. These modifications include trimming the length of the output trajectories, desolvating these trajectories, and removing periodic boundary conditions. Trimming your trajectory allows you to specify the time frame of your simulation that you wish to analyze, desolvating your trajectories will allow you to reduce your file size by excluding water molecules that will not be needed for this analysis, and removing periodic boundary conditions will ensure that your peptide structure is not split across your simulation box boundaries. These steps can be done for trajectory X of your s1 structure using the following command:

gmx\_mpi trjconv -f s1\_prodX.xtc -s s1\_topolX.tpr -o s1\_trimmed\_prodX.xtc -b FIRST\_FRAME -e LAST\_FRAME -pbc mol

\*Note that the units of FIRST\_FRAME and LAST\_FRAME are **picoseconds**, not frame number indices. FIRST\_FRAME will be the frame number you want your trimmed trajectory to start at and LAST\_FRAME will be the frame you want your trimmed trajectory to end at. The -pbc mol flag will remove periodic boundary conditions from your trimmed trajectory.

Ex: if you want to analyze the last 50 ns of a 100 ns simulation, -b would be 50001 and -e 100000.

Repeat this command until you have trimmed trajectory files for every neutral replica of s1 and s2. Then, concatenate the modified trajectories for s1 and s2, respectively, for use in a later step. You can concatenate your trajectory files using the following command:

gmx\_mpi trjcat -f s1\_trimmed\_prod\*.xtc -cat -nosort -o s1cPROT\_all.xtc

\*In this command, all trimmed trajectories for structure s1 will be concatenated. Note that the order in which you concatenate your s1 trajectory files does not matter.

### **1.2: Preparing Files for dPCA Calculations**

**1.2.1: Rewrite “Dummy” GROMACS Trajectories in Terms of Dihedral Angles (VMD\_GenφψIndex.sh and driver\_GenDihedTraj.sh)**

Now that you have created trimmed versions of your trajectory you will need to make a few more modifications before we can begin analysis. Since dPCA calculations are reliant upon calculating the sine and cosine of all *φ* and *ψ* dihedral angles (See Section 2), you will need to rewrite your trimmed trajectories to be written in terms of the backbone dihedral angles of your peptide structure. This can be done by using:

gmx\_mpi angle -f foo.xtc -n dangle.ndx -or dangle.trr -type dihedral

where dangle.ndx (dihedral angle) lists the atom number indices for each *φ* and *ψ* dihedral, line by line. Note that dangle.ndx is an index file that you will need to generate yourself. This can be done by either writing the index file by hand or using a script to generate it from a .gro file. The output dangle.trr is the “dummy” GROMACS trajectory with the sine and cosine of your defined dihedral angles represented as “atom coordinates”. Record your number of dihedrals, *N*. For example, for a cyclic hexapeptide, you may have defined 6 *φ* and 6 *ψ* angles, so 12 dihedrals in total. *N* = 12 is needed for the next step.

**1.2.2: Create “Dummy” Index File and .gro File for “Dummy” GROMACS Trajectory**

Now let’s do a simple math problem:

Given that for every frame in your rewritten “dummy” trajectory dangle.trr, there are 2*N* numbers (1 sine and 1 cosine for each dihedral), and given that in a .gro file, every atom stores three coordinate values, how many “atoms” (*M*) do you have for your *N* dihedrals in every frame?

For a cyclic hexapeptide, *M* = 2\**N*/3 = 8

Remember to round up if *M* is not an integer, so you don’t drop any coordinate. For example, if *N* = 14, *M* = 10.

Here is what’s in the “dummy” index file covar.ndx, enumerating from 1 to *M*. If *M* = 8,

[dummy]

1 2 3 4 5 6 7 8

Then this “dummy” index file is read in along with the “dummy” GROMACS trajectory to create a “dummy” gro file.

gmx\_mpi trjconv -s foo.gro -f foo.gro -o resized.gro -n covar.ndx -e 0

where foo.gro file is the real .gro file for this system (actually, any .gro file would work). The output of this command should be a .gro file with M atoms in it. The actual atoms and coordinate values are irrelevant, because we just need a .gro file with at least 2N coordinate entries. When we pass this .gro file to gmx\_mpi covar, we tell covar to essentially ignore the mass & position data included in this file.

### **Quiz:**

1. What three modifications must be made to your trajectory files? Why must these changes be made?
2. Why must trajectories be rewritten in terms of dihedral angles?
3. Why is a dummy index file created? What is the significance of the number of values listed in this dummy index file?

**2: Performing dPCA Calculation on the “Dummy” GROMACS Trajectory**

### **2.1: Using GROMACS to diagonalize the covariance matrix (phipsi/covar/CalcCovar.sh)**

To actually perform the PCA, you need to diagonalize the covariance matrix of your coordinates. You need to provide your “dummy” gro and index files as well as some options to tell the tool not to do fitting or mass weighting. If something goes wrong here, there might be an error in the files you had to create in the steps before.

gmx\_mpi covar -f dangle.trr -n covar.ndx -ascii -xpm -nofit -nomwa -noref -nopbc -s resized.gro

where dangle.trr is the concatenated dihedral trajectory for both s1 and s2 together. Then in the next step the results will be projected into the same PC space separately for s1 and s2, but using the same eigenvectors from the shared covariance matrix.

The outputs will be eigenval.xvg, eigenvec.trr, covar.log, covar.dat, and covar.xpm.

**2.1.1: Plot the 2D PCs or 3D PCs (phipsi/projection/CalcProject.sh)**

To analyze your PCA results, you can plot 2D or 3D projections of the data onto the eigenvectors of the covariance matrix (the so-called principal components, PCs). To get the free energy landscape along the projections of the first 2 PCs (2D PCs),

gmx\_mpi anaeig -v eigenvec.trr -f dangle.trr -noxvgr -s resized.gro -2d 2dproj\_X\_Y.xvg -first 1 -last 2

For 3D PCs, anaeig does have a -3d flag, so the command below would work:

gmx\_mpi anaeig -v eigenvec.trr -f dangle.trr -noxvgr -s resized.gro -3d 3dproj\_X\_Y\_Z.xvg.gro -first 1 -last 3

Unfortunately, using the -3d flag for anaeig to create PC files does not list values of PC1, PC2, and PC3 in their own respective column. If you wish to create an xvg file that lists all three PCs in their own respective columns, you can run anaeig with the -2d -first 1 -last 2 flag and then run it again with the -2d -first 1 -last 3 flag in order to create two separate 2D PC files that contain the columns: PC1 PC2, and PC1 PC3, respectively:

gmx\_mpi anaeig -v eigenvec.trr -f dangle.trr -xvg none -s resized.gro -2d 3dproj\_X\_Y.xvg -first 1 -last 2

gmx\_mpi anaeig -v eigenvec.trr -f dangle.trr -xvg none -s resized.gro -2d 3dproj\_X\_Z.xvg -first 1 -last 3

Adding the column containing PC3 values to the file containing PC1 and PC2 will then allow you to have an xvg file with three separate columns.

Then you can plot the 2D or 3D PCs using xmgrace or other graphical tool of your choice,

xmgrace 2dproj\_X\_Y.xvg

References (highly recommend you read these!):

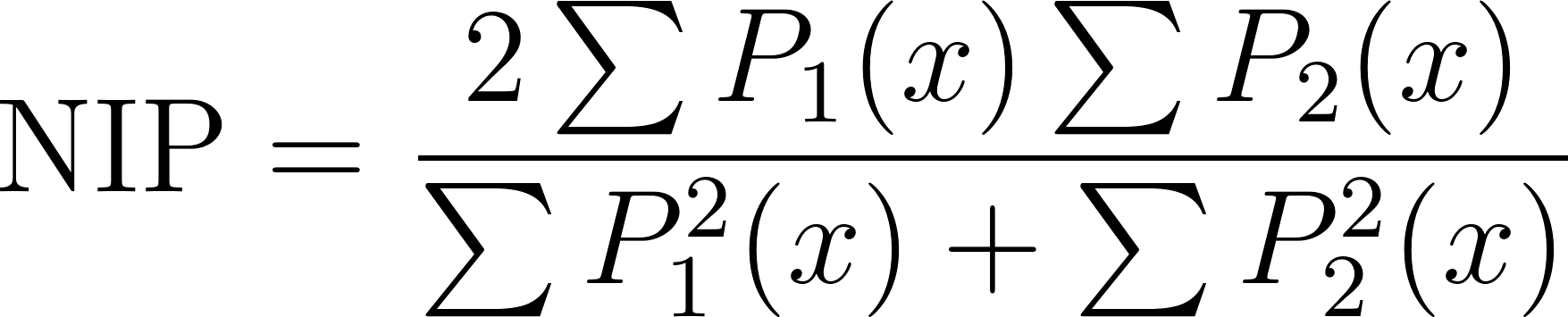
<https://www.researchgate.net/post/Can_anyone_help_me_with_doubts_regarding_dihedral_PCA>

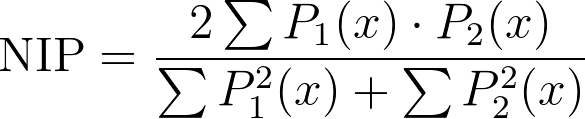
<https://manual.gromacs.org/current/reference-manual/analysis/dihedral-pca.html>

**2.1.2: What Else You Can Do with PCs**

**Normalized Integrated Product (NIP)**

To evaluate the convergence of multiple simulations, NIP is calculated to measure the overlap of the probability density functions (*P*1(***x***) and *P*2(***x***)) of two distribution sets as longer time lengths are included. A value of 1 for NIP means perfect overlap and 0 means no overlap.

[](https://www.codecogs.com/eqnedit.php?latex=%5Ctext%7BNIP%7D%20%3D%20%5Cfrac%7B2%20%5Csum%20P_1(x)%20%5Csum%20P_2(x)%7D%7B%5Csum%20P%5E2_1(x)%20%2B%20%5Csum%20P%5E2_2(x)%7D%20%250)



You can do 2D or 3D histograms on PCs and compare the resulting probability density functions of s1 and s2 with the average of the two for different time windows. In this context, each bin within the 2D or 3D histogram will have a probability density value *P*1(***x***), and you can calculate NIP by using the corresponding bins from a histogram for your second distribution set *P*2(***x***)

A script that calculates 4 NIP values (S1 to S1, S1 to S2, S2 to S1, S2 to S2) starting from the 3D PCA projections may be found in the repository (this script needs Python 2.7.6). Sanity check: The S1 to S1 and S2 to S2 values should both be 1, and the remaining two values should match.

### **Quiz:**

1. What is the difference between using the -3d and -2d flag for the anaeig command? Why might you want to create two different 2d PC files?
2. How can your PCs be used to calculate NIP?
3. What do NIP values tell you?

**3: Using Principal Components for Density-Based Cluster Analysis**

### 3.1: Formatting 3D PC files to perform Density-Based Cluster Analysis (phipsi/projection/driver.sh and phipsi/projection/Py\_combine.py)

Before performing density-based cluster analysis, you will want to ensure that you have text files containing three columns representative of PC1, PC2, and PC3 in your s1 and s2 trajectory. You can double check that you have correctly created this file by using the wc terminal command to check if the number of lines in your 3D PC files match the total number of frames in all of your neutral replicas. (eg. 5 replicas of 100 ns should yield a 3D PC file with 500,000 lines; 5 replicas of 50ns should yield a 3D PC file with 250,000 lines).

Once you have obtained a 3D PC file for your s1 and s2 trajectory, concatenate the two files to create a new .txt file containing the 3D PC information for both structures. You should name these PC files s1PROT.txt, s2PROT.txt, and all.txt (and PROT will be your protein sequence).

### 3.2: Calculating density in the 3D PC space (cluster\_analysis/\*)

NOTE: The following steps of this procedure will be heavily reliant on a variety of scripts. When using these scripts ensure that you are familiar with what each program does and what each output file is used for. These scripts are dependent on Python 2.7.6. This tutorial will give a general explanation for what is happening but it is still important to read through the scripts yourself.

Copy the files from the cluster\_analysis/ directory of the repository to your working directory. Copy the 3D PC .txt files created in step 3.1 to this folder, and run the following command:

python Py\_write\_dPCA\_min\_max.py all.txt PROT “time\_frame” DENSITY

Where you will substitute PROT for your protein sequence, “time\_frame” for the time frame that is being analyzed, and DENSITY for the grid density under which you will not analyze data points. An example of this command would be:

python Py\_write\_dPCA\_min\_max.py all.txt GNSRV “50-100ns” 0.1

This command will generate two files, driver\_s1.sh, and driver\_s2.sh. Each respective bash script will be used to generate the density profile of each respective structure.

Copy the sX/ folder to create a folder called s1/ and a folder called s2/. Then, copy driver\_s1.sh and s1PROT.txt into the s1/ folder, and the corresponding s2 files in the s2/ folder. The following steps will be explained through the s1 folder, but the same procedure will be repeated in the s2 folder.

Bash driver\_s1.sh using the following command:

./driver\_s1.sh

This will generate two .png files that plot the density from your 3D s1PROT.txt files on a 3D grid. One version of the file contains all the points in your s1PROT.txt file, the other will contain a plot of your PC space with lower density points dropped.

The command will also generate a s1PROT.den file, which will list the density of each 3D bin of the 3D PC space in the 4th column of the file. The command will also generate a s1PROT\_kept.dmtx file, which contains the distance matrix that will be used to construct the decision graph that is used when selecting cluster centers.

The wc of some of the files at this point will vary depending on the densities of the different points, but there are a few things to look for.

1. If you head and tail **s\*PROT.den**, you should see four columns. The first three columns give the x,y,z coordinates for the point in PC space, and the fourth column is the density of that point.
2. You should expect the same four columns in the file **s\*PROT\_kept.den**, but this time the lower density points are dropped. So if you wc both files, you should find that **s\*PROT\_kept.den** has fewer lines than **s\*PROT.den**.
3. You can head but especially tail the **s\*PROT\_kept.dmtx** file and should find that the last number in the file should match the number of lines in **s\*PROT\_kept.den.**

### 3.3: Using MATLAB to select cluster centers

To select cluster centers, you will need to use the MATLAB script Mt\_cluster\_dp.m. First open MATLAB, then navigate to the s1/ directory and type:

Mt\_cluster\_dp.m

This will run the script, which will ask you to input the distance matrix file. When prompted with this message type:

s1PROT\_kept.dmtx

A window should pop up, in which you are asked to select cluster centers. The x-axis represents the local density of a given point in your PC space, and the y axis represents the distance to the nearest point of higher density. Generally, select all points above ~0.5 on the y-axis to ensure that cluster centers do not include more than one structure. After selecting points on the graph, exit MATLAB.

This will output a file called CLUSTER\_ASSIGNATION, which assigns an index to each bin in order to separate your data into clusters. The CLUSTER\_ASSIGNATION file should contain the same number of lines as your s\*PROT\_kept.den**.** In this file each grid point has been assigned to a cluster number.

### 3.4: Plotting the clustered 3D PC space and calculating cluster populations

Edit the last lines of driver\_s1.sh by commenting out the lines that say “calc\_den #step1” and “clean #step2”, and uncommenting the line that says “calc\_pop &> populations.txt”

Bash driver\_s1.sh:

./driver\_s1.sh

This should generate a file called populations.txt, along with new .png files that plot the 3D PC space while marking separate clusters by color.

Note: GRID\_ASSIGNATIONand GRID\_ASSIGNATION2 should have the same number of lines as CLUSTER\_ASSIGNATIONand s\*PROT\_kept.den**.** The file GRID\_ASSIGNATIONtakes the grid point coordinates in PC space (the first three columns) and assigns the grid point to a cluster number (fourth column), then the fifth column is the density of that point. GRID\_ASSIGNATION2 is the same, without taking density into account. The file populations.txt will contain the percentage of grid points belonging to each clusters.

Return to your cluster\_analysis/ folder and run the following command:

python Py\_write\_dPCA\_assign\_fortran.py all.txt

This will generate a file called Assign.f90, which will be used to assign frames from the s1PROT.txt file to specific clusters.

Copy Assign.f90 to the s1/struct folder. Then navigate to s1/struct/ and run the following command:

gfortran Assign.f90 -o assign

./assign ../s1PROT.txt assignments.txt

This will generate a file called assignments.txt, in which every line of the s1PROT.txt file has been appended with a cluster index assignment. Note that 0 corresponds to no cluster assignment for that point in PC space.

Note: assignments.txt should have the same length as the total number of frames in all of your neutral replicas . This is because each frame (number in the first column), with its grid points in PC space (next three columns) will be assigned.

### 3.4: Generating the cluster.ndx file

After obtaining the assignments.txt file, bash the driver.sh script that should already be located in your s1/struct directory.

bash driver.sh

This will generate a large index file called cluster.ndx which contains information which assigns every frame of your s1 trajectory to a cluster center. This file will be used in Section 4 to generate cluster trajectories for your s1 structure.

Note: The number of states should be equal to the number of clusters. State 0 corresponds to the points that were dropped because of low density

### **Quiz:**

1. Why is it important to columate your PC files?
2. When using the Py\_write\_dPCA\_min\_max.py script, what are the input values used for?
3. When selecting cluster centers, what do the axes of the graph mean?
4. What information does the assignments.txt file hold?
5. What information is in the cluster.ndx file? What is the file used for?

**4: Generating Cluster Trajectories**

### 4.1: Using your cluster.ndx file to extract frames for each cluster (cluster\_traj/Sh\_make\_cluster\_xtc.sh)

Now that you have obtained a cluster.ndx file for both the s1 and s2 structures. You can use this index file to extract frames from your trajectory that correspond to your top populated clusters. This can be done using the following command:

gmx\_mpi trjconv -f s1cPROT\_all.xtc -s prot.gro -fr cluster.ndx -o s1\_cluster{X}.xtc

In the above command, s1cPROT\_all.xtc is the concatenated raw trajectory file that we created in Step 1.1. Trjconv uses this trajectory in combination with prot.gro, a .gro file containing only your protein structure to extract frames of the concatenated trajectory that correspond to the clusters in cluster.ndx. When prompted by trjconv to select groups of frame number indices, choose the highest group index, since this index corresponds to your most populated cluster.

Then, when prompted to select a group for output select the Protein option (Note: if your .gro file only contains your protein structure, selecting the System option will give the same result). Repeat this process to obtain trajectories for the five most populated clusters of both your s1 and s2 structure.

### **Quiz:**

1. When using the trjconv command, what does the -fr flag mean? How is it used in this section?
2. When prompted to select an index, which index number will correspond to your most populated cluster? Which index corresponds to the least populated cluster?
3. What would be a good way to check that your trajectory does not include any solvent in this step?

**5: Creating Ramachandran Plots**

### 5.1: Preparing cluster trajectories to plotted (rama/driver\_calc\_dihed.sh)

Now that you have created xtc files for the trajectories of your top clusters, you can use these files to create ramachandran plots of your data. You can prepare your trajectory files for plotting by rewriting your xtc files to list the dihedral angles of the residues in your CP sequence using the following command:

gmx\_mpi angle -f s1\_cluster{X}.xtc -n dangle.ndx -type dihedral -all -ov s1\_cluster{X}.xvg

This command will create an xvg file that lists the dihedral angles of all residues in your CP sequence. The format of the data in this file is:

Time (ps) , Average\_Angle, *φ*1, *ψ*1, *φ*2, *ψ*2, … , *φn*, *ψn*

### 5.2: Modifying your xvg files (rama/Clean.sh)

Once you have obtained xvg files for the top clusters of both your s1 and s2 structure, you will need to make one last modification before you can plot your data. Copy each .xvg file as a .txt file, and delete the header of each file. This is done to make your rewritten trajectory files compatible with the script that we will use to plot your data.

### 5.3: Plotting Your Data (rama/calc\_rama.sh and rama/calcFreeEnergy\_v2\_5mers.py)

Copy the above scripts to your directory.

Open the Python script and edit “cvvals=np.loadtxt…” to use the correct number of columns in the cluster{X}.txt file. This python script depends on Python 2.

Eg.) cvvals=np.loadtxt(“...”,usecols=(2,3,4, … , 11), “...”) for a 5mer, or cvvals=np.loadtxt(“...”,usecols=(2,3,4, … , 13), “...”) for a 6mer.

Then, edit the last line of the script, “MakeFigure(10,2, INP, OUT )”. The 10 and 2 refer to the width and height of your plot. For a pentamer, (10,2) works well as input parameters. For a hexamer you will want to change the input to (12,2), (14,2) for a heptamer, etc.

Once you have made these changes, you can finally plot your data. Run the script using the following format:

python calcFreeEnergy\_v2\_5mers.py s1\_cluster{X}.txt s1\_cluster{X}.png

s1\_cluster{X}.txt is the corresponding txt file for the Xth cluster trajectory, and s1\_cluster{X}.png is the .png file that will contain ramachandran plots for all residues in your CP sequence in cluster X.

### **Quiz:**

1. Why are you creating .xvg files from your cluster trajectories? Where are they used?
2. What modifications must be made to the calcFreeEnergy\_v2\_5mers.py script for it to work with your cyclic peptide sequence?

**Quiz Answers**

### Section 1

1. What three modifications must be made to your trajectory files? Why must these changes be made?

The length of your trajectory is trimmed, the trajectory is desolvated, and periodic boundary conditions are removed. These changes are made to ensure that only the specified portion of your protein structure’s trajectory is being analyzed, and that the trajectory that is being analyzed does not include structures split between periodic boundaries

1. Why must trajectories be rewritten in terms of dihedral angles?

Trajectories must be rewritten in terms of dihedral angles because dPCA calculations are reliant upon using the sine and cosine of each φ and ψ angle in the protein backbone.

1. Why is a dummy index file created? What is the significance of the number of values listed in this dummy index file?

The dummy index file is created to generate a dummy gro file that can accomodate the number of parameters that will be used in the dPCA calculation. The equation M = (2\*N)/3 is used because it specifies the amount of lines needed in a gro file to accommodate the sine and cosine values of each dihedral angle in your protein sequence. (3 values per line in the gro file, 2 values per N dihedral angles in your structure).

### Section 2

1. What is the difference between using the -3d and -2d flag for the anaeig command? Why might you want to create two different 2d PC files?

The -3d flag will output three PCs from your analyzed trajectory, while the -2d flag will output two PCs. You will want to create two different 2d PC files in order to create a new 3d PC file that is in a columated format, where each column corresponds to PC1, PC2, PC3, respectively.

1. How can your PCs be used to calculate NIP?

Your PC space can be used to construct a histogram that can be used to measure the population density in each of the bins that are used to construct the histogram. These bin population densities can then be used in the NIP equation to calculate an NIP value in each bin.

1. What do NIP values tell you?

If NIP values for s1 and s2 are pretty far away from 1.0, it is a sign that your simulations are not converged as the structural ensembles of your two runs starting from different conformations are not similar enough. Therefore, check back to restarting/continuing your BE-META simulations in “3.CP: BE-META simulations”.

### Section 3

1. Why is it important to columate your PC files?

The scripts used to calculate density profiles require your PC files to be in this format.

1. When using the Py\_write\_dPCA\_min\_max.py script, what are the input values used for?

all.txt is the file containing the PC1, PC2, and PC3, values for both your s1 and s2 structure, PROT is your protein sequence, time\_frame is the range of time that is being analyzed (note that this value is only used for labeling the generated figures, and does not affect calculations), and DENSITY is the density beneath which points will be removed from the PC space when selecting cluster centroids.

1. When selecting cluster centers, what do the axes of the graph mean?

The x-axis is the local density around a given point, the y-axis is the distance from the point of higher density

1. What information does the assignments.txt file hold?

assignments.txt contains the cluster indices that are assigned to each point in your 3D PC space.

1. What information is in the cluster.ndx file? What is the file used for?

The cluster.ndx file contains a list of the frames of your molecules trajectory that are clustered to specific cluster centers. The most populated cluster is listed last in this file, and the least populated cluster is listed first.

### Section 4

1. When using the trjconv command, what does the -fr flag mean? How is it used in this section?

The -fr flag is used to extract the list frames under a specific directive of your inputted index file. It is used in this section to select the directives within cluster.ndx that contain the frames of the highest populated clusters.

1. When prompted to select an index, which index number will correspond to your most populated cluster? Which index corresponds to the least populated cluster?

The index with the highest values corresponds to the most populated cluster. The index with the lowest value corresponds to the least populated cluster

1. What would be a good way to check that your trajectory does not include any solvent in this step?

When selecting a frame index from the trjconv command, check to see that the number of atoms classified as “System” matches the number of atoms classified as “Protein”

### Section 5

1. Why are you creating .xvg files from your cluster trajectories? Where are they used?

You are creating .xvg files from cluster trajectories so that you can generate a file that explicitly lists the backbone dihedral angles of every residue in your protein sequence. These values can be used by the calcFreeEnergy\_v2\_5mers.py program to create ramachandran plots.

1. What modifications must be made to the calcFreeEnergy\_v2\_5mers.py script for it to work with your cyclic peptide sequence?

The number of columns used in the cvvals=np.loadtxt(“...”,usecols=(2,3,...,N), “...”) must be modified to match the number of dihedral angles in your CP structure. The “MakeFigure(10,2,INP,OUT)” line must also be edited to adjust the width and height of your plot and accomodate for the size of your cyclic peptide structure.