This tutorial series aims to teach you how to create, simulate, and analyze cyclic peptides using gromacs and chimera. It is meant to teach you how and why we complete each step of the process. Although it may be tempting to look at an answer key, it is vital to understand what it is that you are doing so that you are able to expand on it/find issues that may come up. Good luck, have fun, and be safe!

**Building Structure using Chimera**

Before starting this tutorial, you will want to have a little understanding of Chimera. Although you can somewhat google-as-you-go when you forget syntax, it is a good idea to build an understanding of what Chimera is able to do and how to use it. Because Chimera can do some fairly complex processes, it is possible to get overwhelmed by the online resources. While this is somewhat unavoidable, I would recommend going to these two pages for your starting points:

Basic Functions: (Be sure to check out points 1–7, 17, 18, and 19 in the below link). While the command line may seem a little scary compared to the pleasant clickable features, it comes in handy a LOT and can do many things that your mouse can’t do on its own. So be sure not to skip Command Line info!

<https://www.cgl.ucsf.edu/chimera/current/docs/UsersGuide/framecore.html>

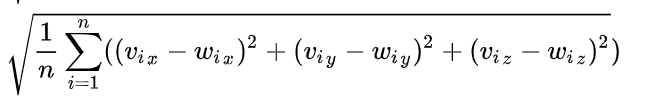
All of the functions/syntax you need, all in one (densely-packed) place! For this tutorial, I would recommend using this tip sheet to find what function you think might help you with a given task, then further googling the function you discovered to get more info on how it works.

<https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/quickref.pdf>

1. To begin, use Build Structure to create an all-Glycine linear peptide. The size of the peptide will depend on how big you want your eventual cyclic peptide to be. A common lab tutorial-sequence is cyclo-(GNSRV), so let’s pretend we’re using that sequence for now. In that case, we want to create a sequence of GGGGG with random phi/psi angles. We’ll eventually add in the actual amino acids that we want in place of the glycines, but Chimera sometimes gets confused when it is asked to cyclize chiral residues and can accidentally change the chirality of the L-/D-amino acids you intend to build. So, for now, all glycines!
2. Once you see your linear peptide, to form a cyclic peptide, delete the OXT atom of the C-terminus and add a bond between the C and N termini (the 5th and 1st residues in this example) using the command line in Chimera.
3. Energy minimize your structure in Chimera (you can use the default options that Chimera provides for charges, etc.). At this point, you should get something that (hopefully) looks somewhat like a cyclic peptide! Typically, all amide bonds in our cyclic peptides are assumed to be in the trans conformation. Unfortunately, during this minimization step in Chimera, a peptide bond might accidentally change into the cis configuration…!!! So, carefully check the structure after energy minimization. If you see a cis bond in your peptide you’ll need to start over and give new random phi/psi angles and try again (sorry!).
4. Now, you’ll want to swap in the actual residues that you want to use in place of the glycines that currently form your cyclic peptide using the command line in Chimera.
5. Lastly, Gromacs expects a given pdb structure not to have any hydrogens. For this reason, we have found that it is best to delete all hydrogens EXCEPT the one connected to the N-terminus of the first residue. The reason for keeping this one hydrogen will be seen later - in summary because the first residue is cyclized, it is necessary to manually insert the carbon as gromacs does not have built-in cyclic peptide support.
6. Lastly, save your structure as a pdb file. After you do this step, you’ll have completed your intro Chimera training!
7. For a given sequence it is necessary to run two simulations, starting from two *different* structures. Then, by comparing these similarity of these simulations over time, you can tell whether you have converged on a solution. So, although you’ve now created one starting structure you’ll need to do steps 1–5 again but using different randomized dihedrals to create your second structure.

**Alignment of the 2 configurations:**

After creating the two (or more) initial confirmations (s1, s2), you can make sure they are substantially different by calculating backbone root-mean-squared-displacement using VMD. The backbone root-mean-squared-displacement (usually simplified to backbone RMSD) is a metric which indicates the separation between two structures’ backbones. In your argon simulation, you likely calculated this metric between individual atoms. The same idea applies here, but we are now adding up the squared displacement between *each and every one* of the atoms that make up the protein backbones of interest in a single time frame. Here is the mathematical way of thinking about this:



Where you have two backbone structures, v and w, each with n atoms.

We first need to load our structures into vmd. To do so, run vmd -m, followed by the names of your two structures:

vmd -m s1.pdb s2.pdb

Then, in vmd, go to **Extension -> Analysis -> RMSD Trajectory tool.** This will give you the ability to calculate your RMSD. Specifically, however, we want to calculate the Backbone-aligned backbone RMSD. This means that we want to first align the protein backbones. This is done by vmd by positioning the two peptides such that their backbones are as close as possible. We are mainly concerned with backbone because the side chains are more variable in simulation (they flop around more), so their initial conformation is much less important than the backbone.

Check the **Backbone** box

Click on **ALIGN**

Then, to calculate the backbone rmsd of the two structures,

Click on **RMSD**

The outputted number will be the RMSD in units of Angstroms.

Typically, you want the structures you come up with to be significantly distinct from each other. If they are not, you’ll need to create another structure (or possibly multiple other structures if they are all too similar). Because RMSD is protein-length dependent (longer proteins->larger rmsd), it is impossible to come up with a single threshold to use. You mainly want to look at the visual alignment of the two structures to determine their similarity at first, but once you start calculating rmsd for many peptides of the same size, you can better understand the range that is possible. Once you have this range, you will be able to set a reasonable threshold below which you’ll want to create another candidate peptide for your simulation. For now though, it’ll be O.K. to simply use your visual prowess to determine if the peptides are too similar. Ask a group member if you are not sure.

**Quiz!**

1. Briefly list the major steps you need to complete to build an initial cyclic peptide in Chimera. It should be possible to outline the procedure in 6-7 steps, but listing too many is better than missing a key step!
2. In Chimera, why do you begin with all glycines then swap in your other residues later?
3. What are the two events that will result in you having to create another candidate peptide?
4. What atoms do you consider when aligning and taking rmsd? (one word answer)
5. What does the RMSD signify and how is it calculated between two protein backbones?

**Generating Initial Gromacs Files**

**Altering Amber99SB Forcefield**

Now that we have our initial pdb structures, we want to convert them into files that gromacs can understand. Unfortunately, gromacs is designed for linear peptides, so we will need to go into the gromacs library to create residues that can be cyclized.

Eventually, we want to be using the RSFF2 force field to simulate our cyclic peptides since RSFF2 is optimized with a protein coil library. This is different from other force fields which aim to predict alpha and beta sheets from (primarily) linear protein libraries. Because cyclic peptides can form a wider range of conformations, it is thus good to optimize on a coil library - one without specific backbone structure. Instead, this force field aims to capture the inherent preference of each amino acid. You should read more in depth about the RSFF2 force field in its paper: “Residue-Specific Force Field Based on Protein Coil Library. RSFF2: Modification of AMBER ff99SB” by Chen-Yang Zhou, Fan Jiang, and Yun-Dong Wu (Google it). In summary, the RSFF2 force field is based off of the Amber99SB force field. A summary of changes made to Amber99SB can be seen in the file in this directory named “0. Rsff2\_vs\_99SB.docx”. Because RSFF2 is derived from Amber99SB, we initially want to create residues in the Amber99SB forcefield that can be cyclized. Then, the Amber99SB topologies we create can be converted to RSFF2 topologies.

To create cyclizable residues,

(1) copy the gromacs library for your current version into a local directory, naming it something like local\_gmx\_library. Then, use the command:

export GMXLIB=/cluster/tufts/(your directory path)

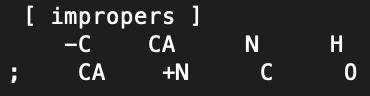
To make sure that gromacs recognizes that you are using this library rather than the default.

(2) make a copy of the Amber99SB force field within your Gromacs force field library and rename it RSFF2\_abc (abc would be your initials). cd into your copied force field RSFF2\_abc and edit the first line of forcefield.doc to distinguish this copied force field from the original.

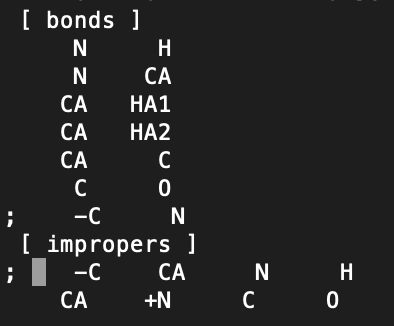
**(3) IMPORTANT!!!!** Now open the forcefield.itp and **change the fudgeQQ parameter to 1.0** instead of 0.833 (0.833 is the default value of in the amber99sb forcefield.itp file).

(4) There are then 3 files that must be altered to allow gromacs to recognize your cyclized residues. Gromacs assumes that any given structure is linear. Assuming that we are again trying to simulation cyclo-(GNSRV), since there “appears” no -C (previous carbon) for the N-terminal GLY and +N for the C-terminal VAL, Gromacs will use NGLY and CVAL for the first and last residues, respectively, and you’ll end up with a topology file for linear GNSRV. (The NGLY and CVAL indicates N-terminal GLY and C-terminal VAL. They can be found in the aminoacid.rtp file in the forcefield directory). To work around this, we can make custom residues CNG and CCV for the “Cyclic N-terminal Glycine” and “Cyclic C-terminal Valine” to trick Gromacs into connecting these two residues and give you a topology file for cyclo-(GNSRV).

* 1. **Make the C-terminal residue cyclizable:** To make the last residue in your sequence cyclizable, first copy the corresponding linear amino acid entry from aminoacids.rtp and paste it in another section in the same aminoacids.rtp file. So, for example, to make a cyclizable valine, we want to start with the information in aminoacids.rtp for VAL (not CVAL!). We do this because the cyclized residue will eventually be the same as its linear counterpart - it should not be interpreted as a C terminal residue. Then, change the name of the residue to, for example, CCV (for Cyclic C-terminal Valine) so Gromacs can distinguish it. Finally, for your new residue, delete the line for the improper dihedral connecting your residue to the N of the next residue (symbolized in gromacs as +N). In the below figure this line is simply commented out for the sake of clarity. This makes it so that Gromacs will not try to connect the 1st residue in your sequence with the last residue – this is the task that gromacs doesn’t know how to do. Instead, this dihedral will have to be manually added in later (as we will see...)

****

* 1. **Make the N-terminal residue cyclizable:** We also want to do a near-identical process for the 1st amino acid in your sequence. The main difference will be that instead of deleting the line that connects your residue’s C-terminal to the next residue, we want to get rid of the lines connecting your residue’s N-terminal to the next residue. To do so, we delete the improper dihedral with the -C atom, as well as the bond between -C and N:



This is necessary because the “-C” symbol stands for the carboxyl carbon of the previous residue. For our N-terminal, there will be no -C. Instead, we will have to manually add a bond between the N-terminal and C-terminal amino acids, as will be seen later.

* 1. Instead, to manually add a bond between the 1st and last residues in your sequence, you want to update specbond.dat so that Gromacs knows how to bond your newly created residues. For this process, type the following at the bottom of the file:Also, make sure to increment the number at the top of the specbond.dat to account for this new bond.

Now, when the command pdb2gmx is run, gromacs will ask whether your want to let it try to treat GLY and VAL with its default method, i.e. as terminal residues (which would be wrong), or if the distance between XXX is < 0.14 nm add a bond between them (which is what we want).

* 1. The last file you need to update is your aminoacids.hdb file. This will tell gromacs how to add hydrogens to the 1st and last residues in your sequence, which are now called CNG and CCV. For this, you want to copy and paste the current .hdb entries for GLY and VAL but rename them for the CNG and CCV residues that you created. For the last residue in your sequence (CCV), this will be sufficient - gromacs should add hydrogens to it just like it would a (non-terminal) linear peptide. For the first residue in your sequence (CNG), however, you will need to:
     1. decrease the number next to the residue name (this number tells gromacs how many lines to expect for that residue) by 1 and
     2. delete the line which adds a hydrogen to its N-terminal (the line containing the -C atom). The hydrogen you left in the pdb of your dehydrogenated peptide will take care of the hydrogen that you are removing from the hdb file (Remember the Building Chimera tutorial). This step would also be unnecessary if gromacs was perfect, but unfortunately this is the solution we have to use for now.

**Generating gro and topology files**

1. To make sure your GMXLIB path is correctly linked, echo $GMXLIB. The path should be to the directory with your altered force field directory. If it is not, enter onto the command line, for example,

export GMXLIB=/cluster/tufts/ylin12/aidan/localGMXlib

1. At this point, you should be able to run your classic pdb2gmx command and create your desired cyclic peptide’s topology and gro files in gromacs. This will take something like the form:

pdb2gmx\_mpi -f GNSRV\_int\_noh.pdb

-o prot.gro

-p cGNSRV\_amber99sb.mod4CPs\_tip3p.top

-ter -inter

-chainsep id -merge interactive

Here, -ter and -inter will generate prompts for you to assign the desired charges to your various residues. “-chainsep id” along with “-merge interactive” will make gromacs recognize that you want a single chain, and, after answering relevant prompts that come up, that you want this chain to be cyclized

* 1. The first prompt will ask you to select your force field. Make sure to select your modified force field. Your force field will be defined by the first line of forcefield.doc (step 1). We want to be able to cyclize our peptide correctly with our altered residues.
  2. The next prompt will ask for the desired representation of water. Choose tip3p, as is recommended for Amber99SB.
  3. There will then be another prompt for you to indicate whether or not you want to bond GLY and VAL using the special bond that you wrote. Type “y” to link your defined starting and ending termini, and “n“ for the remaining prompts.
  4. Finally, if your protein contains residues which can be charged, there will be a prompt asking for their protonation states. For residues such as aspartic acid, choose to create a -1 charge, and for residues such as arginine, choose to create a +1 charge.

This command will also create posre.itp, a file which defines position restraints for heavy atoms in your protein (non hydrogens). This can be included in your top file by using the -DPOSRES flag in the relevant mdp file.

1. With our specbonds.dat file, we have added a bond between the first and last residues in our sequence. However, if we remember back to our alteration of aminoacids.rtp, we also removed key improper dihedrals for the peptide bond between the first and last residues. To do this, we want to manually, open the topology file that was created (.top) and add back the improper dihedrals that you had taken out in step 2a. This should be under [ dihedrals ] function 4. In the case of cyclic GNSRV these are the dihedrals we are adding back. Note: there is a pair of scripts that can complete this for you in called VMD\_GenMissingImpropersForCPs.sh and Py\_AddMissingImpropers.py. However, it is good to do this step manually a couple of times. These files are located in this directory



1. Make sure to check your topology file. Check that your topology file indeed now has (1) a bond between the first and the last residues, and all the associated angles and dihedrals, plus (2) double check there are two improper dihedrals related to the peptide bond between the first and the last residues.

Note that there is a script that can help you with this step of adding our missing improper dihedrals. After you understand the process by hand, this script will save you the time of determining which atom indices need to be included, and will automatically add these improper dihedrals to the topology. The most recent of add\_improper.py should be located in the lab github page, in ysl-lab/Lab-tools/01\_toprep. (https://github.com/ysl-lab/Lab-tools/tree/main/01\_toprep). Be sure to read the README and entire script’s code to understand what it is doing, what its limitations are, and how the script is designed to be used.

**Converting to RSFF2 Force Field**

We want to use the RSFF2 force field, as it captures the behavior of cyclic peptides better than amber99sb. However, RSFF2 is not yet included with gromacs. As a result, to use RSFF2, you’ll need to use a script to transform an amber99sb topology file to an rsff2 topology file.

python g\_mod\_top\_RSFF2\_cyclic.py cGNSRV\_amber99sb\_tip3p.top

cGNSRV\_rsff2\_tip3p.top

If you do not have an RSFF2 folder, you can add one to your own gromacs library. Navigate to your gromacs library and create a new directory called “RSFF2”. Then, copy the forcefield.itp file from your modified amber99sb directory into the RSFF2 folder. Edit the name of the forcefield.itp file to be RSFF2.itp. Then edit RSFF2.itp and make the following changes:

* Modify the fudgeQQ parameter to be 1.0 instead of 0.833
* Modify the #include statements to have a path to the modified amber directory
  + e.g. #include "ffnonbonded.itp" is modified to #include "amber99sb\_cp.ff/ffnonbonded.itp" (to the path back to your modified amber99sb directory)

Note: This script was given by the authors of of the RSFF2 paper mentioned above and then modified by our group members to treat cyclic peptides. There are some discrepancies between the modifications done by this script and those that were described the original RSFF2 force field paper. These discrepancies can be seen in the file in the above directory named “0\_Discrepancies between …”. Unfortunately, the RSFF2 authors are somewhat unclear about the reason for this, but claim that the script only has further optimizations to what was published in the paper.

At this point, you will have a topology file that describes your cyclic peptide with an RSFF2 forcefield! Additionally, you will have your prot.gro file and posre.itp file. These files together describe to gromacs what your cyclic peptide is! (mostly the .top and .gro though)

The most recent version of the g\_mod\_top\_RSFF2.py script should be available from the lab github, in ysl-lab/Lab-tools/01\_toprep. (https://github.com/ysl-lab/Lab-tools/tree/main/01\_toprep). Be sure to read the README and entire code to understand how it functions, what use cases it is built for, and what the expected behavior is in use cases that the script is not prepared to handle. The version of this conversion script as of October 2021 can handle uncapped linear, cyclic, and capped linear peptides (using Ace and NHMe caps). It is important to understand what is different for each of these cases when converting from the Amber99sb to RSFF2 forcefield, and which amino acids have RSFF2 parameterizations.

**Quiz (pt. 2)!**

1. What are the 3 files that must be changed in your gromacs library to create a pair of cyclizable residues?
2. Briefly outline what must be altered in the 3 files you listed
3. Name 3 key differences between RSFF2 and amber99SB
4. What files does pdb2gmx output?