

# Macrocyclic peptide design in Rosetta

This tutorial will guide you through the basics of macrocyclic peptide design based on the protocol published in Mulligan et. al. Computationally designed peptide macrocycle inhibitors of New Delhi metallo-beta-lactamase 1. Proc Natl Acad Sci U S A. 2021 Mar 23;118(12):e2012800118. doi: 10.1073/pnas.2012800118. PMID: 33723038; PMCID: PMC8000195.

## 1. Background

Peptides are interesting molecules because they lie in size between small molecules and have properties of both including high-affinity and incorporation of non-canonical amino acids. Peptides are attractive as protein-protein interaction and enzyme catalysis inhibitors. Mulligan et. al developed a protocol that designs peptide macrocycle inhibitors of New Delhi metallo-beta-lactamase 1, an enzyme that degrades beta-lactam antibiotics. They start from the structure of L-captopril, a small molecule with weak inhibition of New Delhi metallo-beta-lactamase 1 and develop a macrocycle with 50 times greater potency.

The PDB structure of L-captopril bound to New Delhi metallo-beta-lactamase 1 is 4EXS. L-captopril looks a a D-cysteine, L-proline dipeptide and is easily converted into a D-cysteine L-proline dipeptide stub. This stub will serve as the anchor for peptide extension.

Note that example output files for the macrocycle protocol can be found in the `macrocycle/demo` directory.

## 2. Anchor Extension

If you are not already in the macrocycle directory, `cd` to `macrocycle` - assuming you are in `peptide_ncaa_macrocycle_design`:

```
cd macrocycle
```

The prepared inputs for extension can be found in `extend/inputs` and include the dipeptide stub, the Rosetta flags, and a manual foldtree.

Anchor extension requires generation of a foldtree, so that perturbation of anchor stub torsion angles will not disrupt the desired anchor interaction geometry. This can be found in `inputs/foldtree1.txt`. (See Appendix for link on foldtrees.)

From the macrocycle directory:

```
cd extend/inputs
```

Open the dipeptide stub `4EXS_Dcys_Lpro.pdb` in `pymol`:

```
pymol 4EXS_Dcys_Lpro.pdb
```

Find the dipeptide stub that binds to the Zn catalytic site and will serve as the anchor.

Now that you understand where the dipeptide stub binds, we will extend the stub to form an 8 residues macrocyclic peptide. The Rosetta scripts `xml` found in `extend/xml/NDM1i_1_design.xml` uses the `PeptideStubMover` to extend the stub and sets the torsion values of the dipeptide stub backbone to chemically sensible values. Additionally, the `xml` adds peptide cutpoints to N and C terminus and declares a bond between the termini so that Rosetta no longer has repulsive terms for the termini.

Now, move to the `extend` directory:

```
cd ../
```

Make the output directory. Visualize and run the command to extend the peptide stub (should take less than a minute to run):

```
mkdir output
```

```
~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
  -in:file:s inputs/4EXS_Dcys_Lpro.pdb -parser:protocol xml/NDM1i_1_design.xml \
  @inputs/rosetta.flags -out:prefix output/ -nstruct 5
```

This command creates five extended peptides that can be found in the output directory. Visualize these structures in pymol:

```
pymol output/*.pdb
```

Notice that the geometry of the N to C terminal bond is incorrect because we have not yet closed the bond with loop closure (genkic), only declared that it exists.

### 3. Cyclization

In Rosetta, Generalized Kinematic Closure (GenKIC) is used to close/model loops and can go through covalent linkages such as disulfide bonds of N to C terminal cyclization. We will use genkic to close the terminal peptide bond, setting the angles and bond distances of this bond to the ideal value. Additionally, genkic is used to sample different backbone geometries of cyclic peptides that can be designed. The XML in `cyclize/xml/NDM1i_1_design.xml` closes the terminal peptide bond and filters for peptide internal hydrogen bonds - important for designing stable peptides that lack a hydrophobic core - and steric clashes with the receptor. This step is computationally expensive due to the high filter failure rate, so you should open a new terminal tab to run these commands and look at the results later.

From the macrocycle directory in a new tab (should take about 2 minutes for 5 backbones - all may not be successful):

```
cd ../cyclize
mkdir output

~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
  -in:file:s inputs/4EXS_Dcys_Lpro.pdb -parser:protocol xml/NDM1i_1_design.xml \
  @inputs/rosetta.flags -out:prefix output/ -nstruct 5
```

Once the backbone cyclization completes, you can use these backbones as the starting point for design, but note that some designs are expected to fail filters. In an actual peptide search, you would generate thousands of designed peptide from different backbones. For now, move onto designing a given backbone in the design directory.

### 3. Design

The design protocol for cyclic peptides uses a combination of repacking and minimization to design a given backbone and filters for oversaturated hydrogen bond acceptors - the Rosetta energy function is pairwise and cannot detect oversaturated hydrogen bond acceptors - as well as shape complementarity and internal hydrogen bonds. The packer palette for design includes the L amino acids and their D stereoisomers, but excludes GLY and CYS residues to aid with conformational stability of the design.

Additionally, we can include non-canonical amino acids, such as TFF from Part 1 of the tutorial in the design palette.

Note that this protocol uses amino acid composition constraints to enforce among other things incorporation of hydrophobic and proline amino acids.

Edit the XML script to add TFF to the set of residues being designed:

```
cd ../design
gedit xml/NDM1i_1_design.xml
```

While editing the XML script, add TFF to the packer line, the line for L hydrophobic design (only needed because of the amino acid composition used - see Appendix), and write the change. Lines similar to the following should already exist in the XML - find them and edit in the changes

Under `PACKER_PALETTES`:

```
<CustomBaseTypePackerPalette name="design_palette"
  additional_residue_types="DALA,DASP,DGLU,DPHE,DHIS,DILE,DLYS,DLEU,DMET,DASN,DPRO,DGLN,
    DARG,DSER,DTHR,DVAL,DTRP,DTYR,TFF"
/>
```

Under TASK\_OPERATIONS:

```
<RestrictToSpecifiedBaseResidueTypes name="L_hydrophobic_design"
  base_types="PHE,ILE,LEU,MET,PRO,VAL,TRP,TYR,TFF"
  selector="select_L_hydrophobic_positions"
/>
```

We will be designing with provided backbones that can be found in `inputs/4EXS_Dcys_Lpro_native.pdb` and `../cyclize/output/*.pdb`. The PDB in inputs is the backbone of one of the crystalized macrocycle inhibitors and the demo backbones are provided to ensure higher probability of successful design and incorporation of TFF.

To design with the given backbone (Will take about 20 minutes, but you can start the visualization as they are made approximately every 2 minutes):

```
mkdir output

~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
  -in:file:s inputs/4EXS_Dcys_Lpro_native.pdb ../demo/cyclize/output/*.pdb \
  -in:file:extra_res_fa ../../ncaa/output_files/TFF.params \
  -parser:protocol xml/NDM1i_1_design.xml \
  @inputs/rosetta.flags -out:prefix output/ -nstruct 1
```

Visualize these structures in pymol:

```
pymol output/*.pdb
```

## Optional Monte Carlo Perturbation of Initial Designed Peptides

Mulligan et. al. used a monte carlo protocol to explore the local conformational space of initial designed peptides, optimizing the peptide - enzyme shape complementarity. The xml for this step is more complicated and the procedure is computationally expensive, so this section is optional to run and would require a deeper dive to fully understand.

To run the monte carlo protocol:

```
cd ../mc_sample
mkdir output

~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
  -in:file:s inputs/4EXS_Dcys_Lpro_native_0001.pdb -parser:protocol xml/NDM1i_1_design.xml \
  @inputs/rosetta.flags -out:prefix output/ -nstruct 5
```

## Full Protocol

The full protocol, combining all steps into one xml can be found in the `og_scripts` directory

## Appendix

For the original github for these scripts: [https://github.com/vmullig/ndm1\\_design\\_scripts](https://github.com/vmullig/ndm1_design_scripts)

For more details on foldtrees, go here: [https://docs.rosettacommons.org/demos/latest/tutorials/fold\\_tree/fold\\_tree](https://docs.rosettacommons.org/demos/latest/tutorials/fold_tree/fold_tree)

Other helpful movers for modeling peptides in Rosetta:

CycpepRigidBodyPermutationMover [https://docs.rosettacommons.org/docs/latest/scripting\\_documentation/RosettaScripts/Movers/movers\\_pages/CycpepRigidBodyPermutationMover](https://docs.rosettacommons.org/docs/latest/scripting_documentation/RosettaScripts/Movers/movers_pages/CycpepRigidBodyPermutationMover)

Simple Cyclic Peptide Prediction (simple\_cycpep\_predict) Application [https://docs.rosettacommons.org/docs/latest/structure\\_prediction/simple\\_cycpep\\_predict](https://docs.rosettacommons.org/docs/latest/structure_prediction/simple_cycpep_predict)

PeptideCyclizeMover [https://docs.rosettacommons.org/docs/latest/scripting\\_documentation/RosettaScripts/Movers/movers\\_pages/PeptideCyclizeMover](https://docs.rosettacommons.org/docs/latest/scripting_documentation/RosettaScripts/Movers/movers_pages/PeptideCyclizeMover)

Amino Acid Composition:

- [https://docs.rosettacommons.org/docs/latest/scripting\\_documentation/RosettaScripts/Movers/movers\\_pages/AddCompositionConstraintMover](https://docs.rosettacommons.org/docs/latest/scripting_documentation/RosettaScripts/Movers/movers_pages/AddCompositionConstraintMover)
- [https://docs.rosettacommons.org/docs/latest/rosetta\\_basics/scoring/AACompositionEnergy](https://docs.rosettacommons.org/docs/latest/rosetta_basics/scoring/AACompositionEnergy)