

Cyclic Peptide AI Design Tutorial

This tutorial will guide you through the complete workflow for designing cyclic peptides using AI tools. We will first generate cyclic backbones with RFDiffusion, then design the backbone sequence with ProteinMPNN, and finally predict and validate the peptide structure with AlphaFold2.

Step 1: Backbone Generation with RFDiffusion

We begin our design process by generating cyclic peptide backbones using RFDiffusion, which has been trained to understand protein geometry and can generate realistic backbone conformations. For cyclic peptide generation with RFDiffusion, peptides of sizes 10-14 are generally good sizes to use, though you can experiment with smaller or larger peptides as well. For an individual peptide, the RFDiffusion CPU code generally takes less than half a minute per structure on the molgraph machines, so feel free to increase the `num_designs` parameter. The benefit of staying at or under 14 amino acids is that you can re-predict the peptide structure using the orthogonal Rosetta `simple_cyc_pep_predict` application.

To generate the cyclic peptide backbones, activate the RFDiffusion environment using `conda activate SE3nv` and create the output directory with `mkdir -p cyclic_gen`. Then run the following command:

```
~/rosetta_workshop/RFDiffusion/scripts/run_inference.py \  
  'contigmap.contigs=[10-10]' \  
  inference.output_prefix=./cyclic_gen/macrocycle \  
  inference.num_designs=5 \  
  inference.cyclic=True \  
  inference.cyc_chains='a'
```

This command generates five different 10-amino acid cyclic peptide backbones, with the cyclic constraint applied to chain 'a'. The `contigs` parameter `[10-10]` specifies that we want peptides of exactly 10 residues, but you can modify this to generate different sizes such as `[8-8]`, `[12-12]`, or even ranges like `[10-14]`.

Workshop participants should feel free to change the peptide size in the `contigs` or design more peptides by increasing the `num_designs` parameter. The generated backbones will be saved to the `cyclic_gen` directory. When examining these structures, look for backbones that display clear secondary structure elements such as beta sheets or alpha helical character, as these are more likely to lead to successful designs for the next steps.

Step 2: Sequence Design with ProteinMPNN

Now, we will use ProteinMPNN to design your favorite peptide backbone from step 1. When making this selection, prioritize peptides with more internal hydrogen bonds and, those that have more beta sheet or alpha helical character, as these structural features are more likely to be successful for generating a stable peptide with this pipeline.

To begin sequence design, activate the ProteinMPNN environment with `conda activate ligandmpnn_env` and create the output directory using `mkdir -p mpnn_design`. Then run ProteinMPNN with the following command:

```
python ~/rosetta_workshop/LigandMPNN/run.py \  
  --checkpoint_protein_mpnn ~/rosetta_workshop/LigandMPNN/model_params/proteinmpnn_v_48_020.pt \  
  --checkpoint_path_sc ~/rosetta_workshop/LigandMPNN/model_params/ligandmpnn_sc_v_32_002_16.pt \  
  --pdb_path cyclic_gen/macrocycle_1.pdb \  
  --out_folder mpnn_design/ \  
  --model_type "protein_mpnn" \  
  --number_of_batches 1 \  
  --batch_size 10 \  
  --pack_side_chains 1 \  
  --number_of_packs_per_design 1
```

Remember to update the `--pdb_path` parameter to point to your chosen backbone structure from the `cyclic_gen` directory. This command will generate 10 different sequence designs for your chosen backbone, with optimized side chain conformations. The `--pack_side_chains 1` option ensures that ProteinMPNN not only designs the sequence but also optimizes the side chain rotamers for better packing. The packed designs can be visualized using `pymol mpnn_design/packed/*.pdb`, where you can examine how well the designed sequences fill the backbone structure. The designed sequences themselves can be found by examining the contents with `cat mpnn_design/seqs/macrocycle_*.fa`, which contains the amino acid sequences in FASTA format.

Step 3: Structure Prediction with AlphaFold2

The final step in our workflow involves using AlphaFold2 via ColabDesign to predict the structure of our designed cyclic peptide. This structure prediction step will validate whether AlphaFold thinks our design will fold as intended. To predict the structure, we will use AlphaFold with a cyclic peptide offset via a Python script that uses ColabDesign to predict the structure of the designed cyclic peptide.

Before running the AlphaFold prediction, set the JAX platforms to CPU using `export JAX_PLATFORMS=cpu` as the molgraph machines do not have a GPU available. Extract your chosen designed sequence from the ProteinMPNN output by examining the sequences in the `mpnn_design/seqs` directory. Once you've selected a promising sequence, run the AlphaFold prediction using the provided script and the `colabdesign python` from `sbgrid`:

```
python.colabdesign af_cyc_pred.py KILPGVISEG -o af_output --model_num 1 \
  --params_path /programs/x86_64-linux/colabdesign/1.1.2/
```

Replace “KILPGVISEG” with your actual designed sequence. The script automatically applies cyclic peptide constraints by connecting the N-terminus to the C-terminus, ensuring that AlphaFold considers the cyclic nature of the peptide during structure prediction.

The resulting AlphaFold predicted peptide can be visualized at the default location using `pymol af_output/cyclic_peptide.pdb`. In PyMOL, the `spectrum b` command will color the peptide by B-factor, which corresponds to AlphaFold's confidence scores. The range of B-factors is printed to the screen. The script also generates a results file containing key metrics including pLDDT (overall confidence), pTM (template modeling score), and PAE (predicted aligned error), which help assess the quality and reliability of the prediction. For peptides, pLDDT is generally the best method to assess quality and pLDDT scores above 0.9 (this pLDDT is rescaled to be between 0 and 1) are considered good. Because peptides are so small, the pTM is not an accurate representation of a peptide's stability.

Expanding this Workflow to Peptide Binder Design

This workflow can easily be expanded to designing peptide binders either de novo or that stabilize a known binding motif. In this use case, you may want to generate 10,000+ starting backbones for sequence design. After designing the peptide sequence and predicting its structure in complex with the target, iteratively redesigning and repredicting the peptide-target complex can optimize the peptide-target interactions into a design minima. The ColabDesign github (<https://github.com/sokrypton/ColabDesign>) has a `designability_test.py` script that can serve as a template for designing and validating peptide binders.

Alternatively, structure prediction methods based on AlphaFold3 such as `boltz` and `RosettaFold3` can be used to validate peptide-protein complexes.