Report for Assignment 3 - Serial Ab Initio Protein Folding Using PyRosetta

Name : Navaneeth Shaji

Roll no: 21CS30032

Structure Prediction

The chosen method for the structure prediction is the 'Ab initiio folding method' .

The Villin headpiece sequence chosen is "MLSDEDFKAVFGMTRSAFANLPLWKQQNLKKEKGLF" which is the 36 AA sequence chosen from RCSB.

Parameters Chosen

Monte Carlo parameters : KT = 3.0, cycles = 50000

Functions used

1. pyrosetta.init("-in::file::fullatom -mute all") : Initializes the PyRosetta environment.
2. pyrosetta.pose_from_sequence(sequence,res_type) : Returns a Pose object generated from a single-letter sequence of amino acid residues in sequence using the res_type. res_type = 'fa_standard' will give a full atom pose of the sequence.
3. pose.total_residue() : returns the total number of the residues in the sequence.
4. pose.set_phi(i, -150) : sets the phi torsion value of the residue to -150.
5. pose.set_psi(i, 150) : sets the psi torsion value of the residue to 150.
6. pose.set_omega(i, 180) : sets the omega torsion value to 180.
7. rosetta.protocols.simple_moves.SwitchResidueTypeSetMover(res_type) : Toggles between centroid and full atom modes. If used with res_type='centroid' , it converts the full atom pose to centroid representation where sidechains are reduced to simple atoms.
8. rosetta.core.kinematics.MoveMap() : creates a movemap object which defines which part of the protein can move and how they can move during simulation.
9. movemap.set_bb(True) : allows the backbone atoms of the protein sequence to move.
10. rosetta.core.fragment.ConstantLengthFragSet(frag_len,frag_file) : Reads the provided fragment library file and creates a set of fragments of a given constant length.
11. rosetta.protocols.simple_moves.ClassicFragmentMover(frag, movemap) : used to apply fragments from the fragment set to the pose based on the MoveMap
12. rosetta.protocols.moves.RepeatMover(fragmover9, 3): used to to repeat the application of a mover
13. rosetta.protocols.moves.SequenceMover() : creates a sequencemoverobject which allows for the application of multiple movers in the sequence.
14. rosetta.core.scoring.ScoreFunctionFactory.create_score_function("score3") : creates a ScoreFunction object that uses the `"score3"` scoring method and can be used to calculate the energy of the pose.
15. rosetta.protocols.moves.MonteCarlo(pose, score_fun, kT) : creates the main object for the Monte-Carlo simulations.
16. rosetta.protocols.moves.TrialMover(seq_mover, mc) : wraps the sequence mover and after each iteration , accepts or rejects depending on the Monte-Carlo object.

17. rosetta.protocols.moves.RepeatMover.apply(pose) : applies the RepeatMover to the pose. The simulation runs for the specified number of iterations.
18. rosetta.protocols.moves.MonteCarlo.recover_low(pose) : copies the lowest scores pose into the input pose object.
19. pose.dump_pdb(filename) : writes the final full atom pose to a PDB file.

Comparison on template based and Ab initio method

Template-based modeling uses an existing structure (template) of a protein that is homologous (similar in sequence) to the target protein. This method relies on the assumption that homologous proteins will have similar structures. It is highly accurate when a suitable template is available and allows for faster predictions since it leverages known structural information, making it computationally efficient and less resource-intensive.

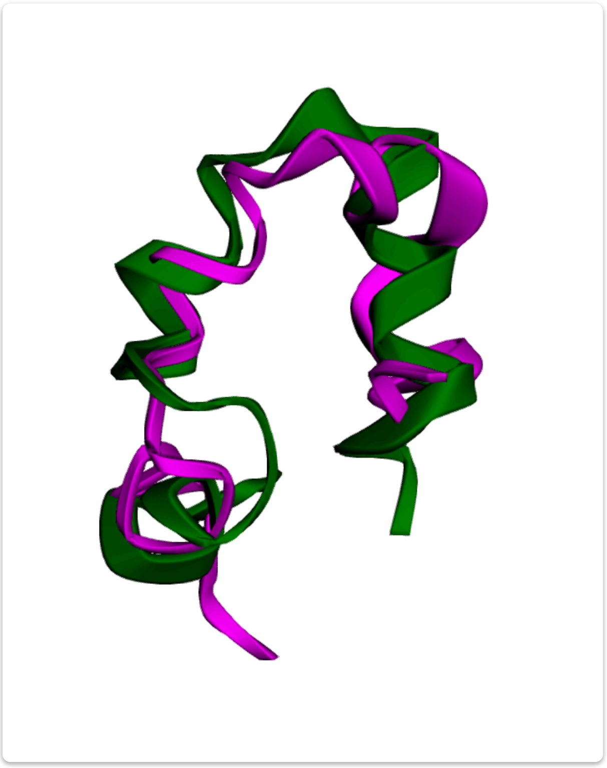
Ab Initio Protein Folding, on the other hand, predicts a protein's structure from scratch, based solely on its amino acid sequence, without the need for a homologous template. This method explores the conformational space through sampling and energy minimization techniques to find the most stable structure. While it is versatile and can be used for any protein sequence, it is computationally expensive, slower, and typically less accurate, especially for larger proteins, as it does not rely on pre-existing structural knowledge.

Results from Ab-initio method

```
Number of C-alpha atoms in native structure: 36
Number of C-alpha atoms in predicted structure: 36
RMSD between predicted and native structure: 5.258 Å
Final Pose energy: 30.851761539044897
```

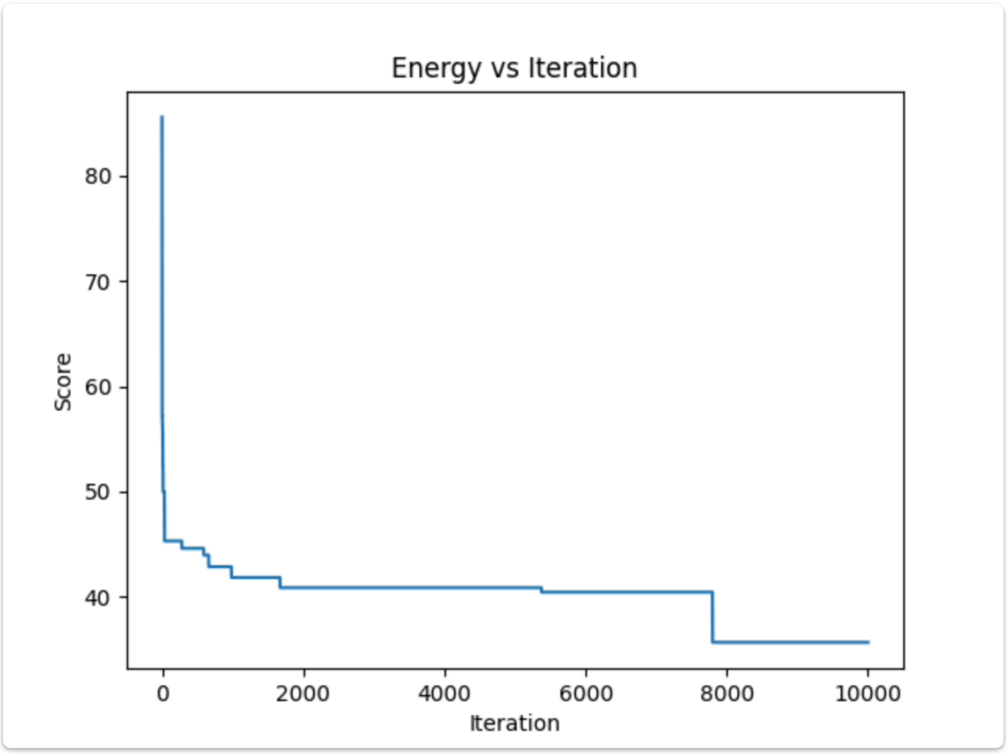
Predicted Structure

native structure - green
predicted structure - magenta



Energy results

The following is the energy variation over the iterations :



Steps to run the code

First ensure all the required libraries are install

```
pip install pyrosetta-installer
python -c 'import pyrosetta_installer; pyrosetta_installer.install_pyrosetta()'

pip install biopython py3Dmol matplotlib requests
```

The main file is main.py file . To run it ,

```
python3 main.py
```

for the visualisation, run the alignment.ipynb file
For this ensure that you have kernel with the required libraries.

References

Links to the fragment files : aat000_09.frag aat000_03.frag
Link to the native pdb file : native.pdb
Pyrosetta documentations used :
https://graylab.jhu.edu/PyRosetta.documentation/pyrosetta.html
https://docs.rosettacommons.org/manuals/archive/rosetta_3.6_user_guide/core+protocols/d8/d0b/classcore_1_1kinematics_1_1_move_map.html