# Assignment 2 – Atomic-Level Molecular Modeling

CS/BIOE/CME/BIOPHYS/BIOMEDIN 279

Due: October 21, 2021 at 3:00 PM

The goal of this assignment is to understand the biological and computational aspects of macromolecular energy functions and of predicting the three-dimensional structure of a folded protein.

**Acknowledgements:** Portions of this assignment are based off of the PyRosetta Tutorials.

## 1 Preliminaries

- For this assignment we **strongly recommend using LTS machines** for the coding portion of the assignment. Installing PyRosetta correctly on Windows can be very challenging, and we are unable to provide support with this process on Windows. Mac and Linux users have a better chance of success with self-installation, but it has proven difficult for many students in the past. See the Assignment 2 Setup Document for details on using PyRosetta on LTS, and information on self-installation for Mac and Linux users.
- **LTS** users make sure to use the command "python3.8" instead of "python".
- We request that students living on campus utilize in-person LTS clusters. Students who live off campus and are not on campus frequently can use remotely accessible LTS machines. The supply of remotely accessible LTS machines is limited, so you may not be able to access one just before the assignment deadline; please plan accordingly.
- Please start the assignment early so that we have time to address any software challenges in office hours and via Ed well before the deadline.

## 2 Force Fields and Free Energy Calculation

### 2.1 Potential Energy and Force Fields

Molecules, including proteins and other biomolecules, prefer to occupy energetically stable (i.e. low energy) three-dimensional structures. Thus, in order to predict which conformations a protein or other biomolecule is likely to adopt, it's important to understand the forces acting between its atoms.

One frequently used type of potential energy function for biomolecules, called a molecular mechanics force field, is typically defined as a sum of bond length, bond angle, dihedral angle, electrostatics, and van der Waals terms (see slides from the lecture on "Energy Functions  Their Relationship to Molecular Conformation," particularly slide 19).

$$U(c) = U_b(c) + U_\theta(c) + U_\phi(c) + U_e(c) + U_v(c)$$

**Note:** $U_b(c)$: bond length stretching, $U_\theta(c)$: bond angle bending, $U_\phi(c)$: torsional angle twisting, $U_e(c)$: electrostatic interaction, $U_v(c)$: van der Waals interaction.

**Question 1:** *For a potential energy function defined by such a molecular mechanics force field, which terms (bond angle, bond length, dihedral angle, electrostatics, or van der Waals) will require the most computational time? (Assume the molecular system being simulated contains a very large number of atoms.)*

**Question 2:** *Structural biologists often refer to the "hydrophobic effect" as a determinant of protein structure. This "effect" is said to cause certain regions of a protein to fold such that they are no longer exposed to water. Explain what drives this "effect" in terms of the electrostatic forces involved.(A brief, simple explanation is sufficient.)*

## 2.2 Free Energy

Recall that proteins and other biomolecules are constantly changing shape as their atoms move around. Thus, when we talk about protein structure prediction, we're really interested in finding a set of conformations where the protein will spend most of its time.

Let's consider some set $C$ of similar conformations. (We're defining an individual conformation $c$ as a particular arrangement of all the atoms, as specified by precise coordinates for each atom.) We frequently refer to such a set of conformations as a macrostate (or a conformational state, if the set of conformations corresponds to a well in the potential energy surface). We know the probability that a bimolecule $b$ is in some conformation $c$ is Boltzmann distributed according to $U(c)$, where $U(c)$ is the potential energy associated with conformation $c$. This means we can express the probability that the biomolecule will adopt some conformation in the macrostate $C$ as the following sum.

$$\Pr\left(b \text{ adopts some } c \in C\right) \propto \sum_{c \in C} e^{-U(c)/(k_B \cdot T)}$$

$k_B$*: Boltzmann constant*
$T$*: temperature*

Intuitively, a macrostate is well-populated if the energy of each conformation in the macrostate is sufficiently low and the number of conformations in the macrostate is sufficiently large. We can define the free energy of the macrostate $C$ as the value $G_C$ that satisfies

$$\Pr\left(p \text{ adopts some } c \in C\right) = e^{-G_C/(k_B \cdot T)}$$

It is important to understand the distinction between minimizing potential energy and minimizing free energy: minimizing potential energy gives the single most stable conformation ($c$) while minimizing free energy gives the most likely macrostate ($C$).

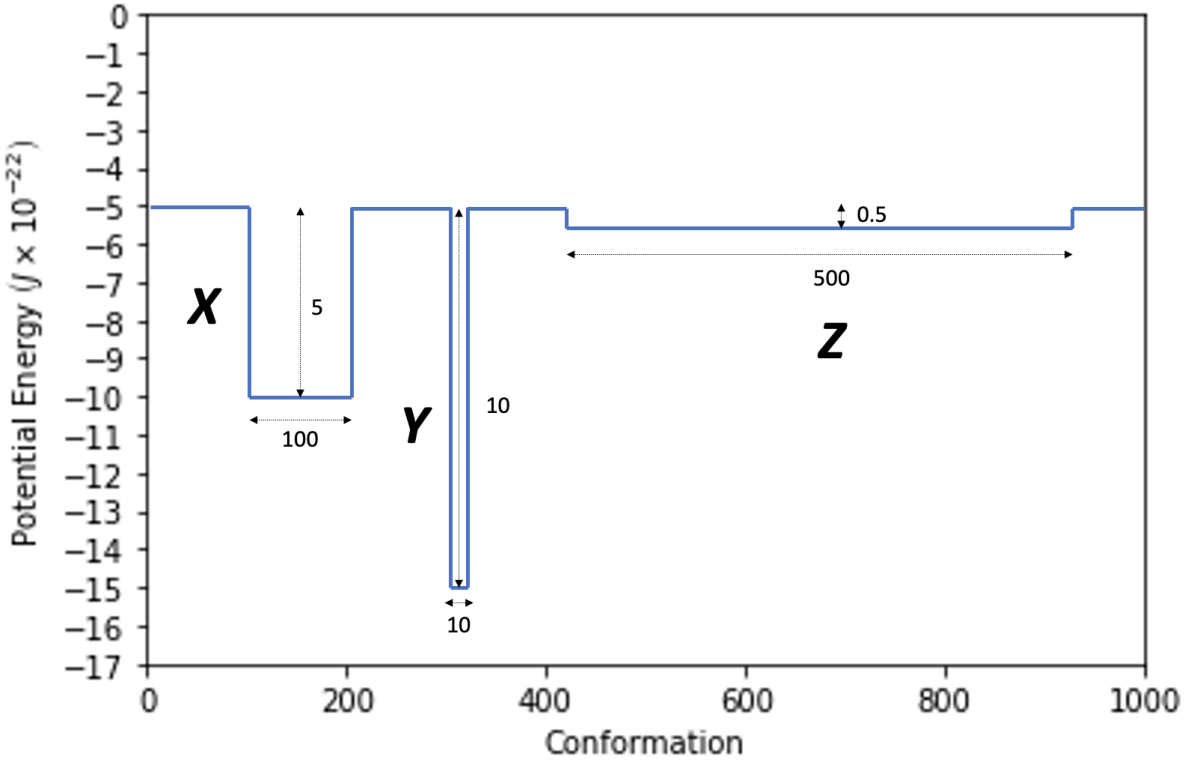


Figure 1: Potential Energy vs. Conformation

**Question 3:** *Consider Figure 1. Let's say this plot represents the potential energy of a given protein (and the water surrounding it) as it adopts different conformations.*

*(a) Intuitively, which conformational macrostate ($X, Y$, or $Z$) would you expect to be preferred at very low temperatures? Which conformational macrostate ($X, Y$, or $Z$) would you expect to be preferred at very high temperatures?*

*(b) Now, calculate the approximate relative percentage present of each macrostate at each of the temperatures: 1 K, 100 K, and 310 K by using the free energy equation. Display your results in a table as shown below. Note that 310 K (equivalent to 37 °C or 98.6 °F) is normal human body temperature.*

| | Percentage Present of each Macrostate (%) | | |
|---|---|---|---|
| **Temperature (K)** | **X** | **Y** | **Z** |
| *1* | | | |
| *100* | | | |
| *310* | | | |

*Hints:* $k_B = 1.38 \times 10^{-23}$J/K. *You may ignore the time the protein spends outside of macrostates X, Y, and Z. Pay close attention to the sign of the potential energy.*

*(c) Did the calculated preferred macrostate at 1 K and 310 K match your initial intuition? Why or why not? Make sure to include a discussion of temperature and the potential energy function in your answer.*

This question should illustrate that the optimal structure of a protein (or any molecule for that matter) varies with temperature.

## 2.3 Knowledge-Based Force Fields

In the following exercises, we will be using PyRosetta which utilizes the Rosetta Force Field or Score Function. One key design decision behind Rosetta is the use of a knowledge-based force field. Instead of trying to approximate the potential energy of atomic interactions directly, this force field tries to estimate the free energy of structures by incorporating knowledge of solved protein structures. In other words, conformations are deemed more likely, or lower "energy", if they are similar to conformations that are known to exist in other proteins.

**Question 4:** *What is one limitation of knowledge-based protein structure prediction methods?*

# 3 Structure Prediction

**Question 5:** *Suppose we have a well-specified free energy function $G(c)$ that provides a very accurate free energy estimate for an ensemble (macrostate) of conformations similar to conformation $c$ of a protein. Since we want to determine the conformation $c$ for which $G(c)$ has lowest free energy, what prevents us, practically, from evaluating $G(c)$ for all reasonable $c$ to determine which macrostate has the lowest free energy?*

## 3.1 Monte Carlo Methods

In the context of biomolecular simulation and structure prediction, the Metropolis **Monte Carlo** algorithm goes as follows.

1. $c \leftarrow$ select an initial conformation
2. Repeat:
   - $c' \leftarrow$ sample a local random move from $c$ (e.g. change one dihedral angle)
   - if $U(c') \leq U(c)$,

$$\text{assign } c \leftarrow c'$$

else

$$\begin{cases} \text{assign } c \leftarrow c' & \text{with probability } e^{-\Delta U/(k_B \cdot T)} \\ \text{do nothing} & \text{with probability } 1 - e^{-\Delta U/(k_B \cdot T)} \end{cases}$$

In our implementation, we will use this algorithm. Note that the proposed move is accepted if the energy decreases, and is accepted with exponentially decreasing probability if the energy increases. This acceptance profile is called the Metropolis Criterion.

**Question 6:** *If our goal is to find conformations with low free energy, why would we ever want to accept a structure with higher energy than our current estimate?*

**Question 7:**

*(a) Implement* `acceptMove()` *in* `MonteCarloPredictor` *in* ***predictor.py****.*
`acceptMove()` *should take in the current energy and the energy of the proposed move, and return* `True` *according to the Metropolis Criterion.*

*(b) Implement* `predict()` *in* `MonteCarloPredictor` *in* ***predictor.py****.*
*Implement the Monte Carlo sampling algorithm with the Metropolis Criterion described above. Make calls to* `sampleMove()` *and* `acceptMove()` *where appropriate.*

***Note:*** *only write and modify code in the lines bound by the following comments*

```
### BEGIN YOUR CODE HERE ###

### END YOUR CODE HERE ###
```

Now that you've implemented the basic framework for the Monte Carlo / Metropolis algorithm, we can start to think about what types of perturbations to the current structure we should explore. For the rest of the assignment we will be exploring different methods that we can use (with varying degrees of success) to predict the secondary structures of hras from **pdbs/sequence.pdb**. First, we will try purely random changes in the dihedral angles.

PyRosetta provides support to interface with PyMOL, so that you can visualize what is happening to a given protein as it is exploring the conformational space. We must initialize PyMOL to listen for calls from PyRosetta. To do this, in PyMOL, run the python script provided in the `assn2` directory.

```
PyMOL> run PyMOLRosettaServer.py
```

**Question 8:**
`sampleMove()` *in the* `DihedralPredictor` *class has been implemented for you. This function takes in a* `Pose` *object, and returns a new* `Pose` *object which has sampled one of three types of dihedral moves.*

*Specifically, we select a residue at random, then choose one of the following three moves at random: (1) a dihedral move in $\phi$, (2) a dihedral move in $\psi$, or (3) a shear move, where you select a $\Delta\phi$, and then update $\psi$ by $-\Delta\phi$. A move is defined as a rotation of the dihedral angle.*

*Changes in dihedral angle will be sampled from a normal distribution centered at 0 with standard deviation 5°. It's a good idea to take a look at the code to get a sense of what is happening in this function call.*

*(a) Run* `DihedralPredictor` *for 1000 iterations on* `sequence.pdb` ***in terminal*** *and watch it search the conformational space in PyMOL. The predicted structure will be saved in the "out" folder under the name "dihedral1000.pdb".*

```
python3.8 predict.py pdbs/sequence.pdb dihedral1000.pdb -dihedral
-pymol -1000
```

*Note that the first argument "pdbs/sequence.pdb" specifies the input pdb filename and the second argument "dihedral1000.pdb" specifies the output pdb filename.*

*After 1000 iterations, generate a Ramachandran plot using the included script* `ramaPlot.py` *from the terminal. <file1> refers to the filename of your predicted pdb structure.*

```
python3.8 ramaPlot.py out/<file1>.pdb
```

***Include a screen-shot of both the structure and the Ramachandran plot*** *in your writeup.*

*(b) Now run the predictor for 100,000 iterations (remove the -pymol flag for the sake of speed).*

```
python3.8 predict.py pdbs/sequence.pdb dihedral100K.pdb -dihedral
-100000
```

*Include a screen-shot of the structure and of the Ramachandran plot.*

*(c) Compare the structure and Ramachandran plots observed after 1000 and 100,000 iterations. Does the structure resemble a folded protein to a greater degree after increasing the number of iterations? Can you distinguish any distinctive secondary structures in PyMOL or in the Ramachandran plots?*

**Question 9:** *As the structure changes according to the algorithm, should we expect it to follow the pathway by which the real protein folds? Why or why not?*

While perturbing the structure one dihedral angle at a time seems like a natural way to sample the search space, you can see that it will take a lot of random samples to converge to anything that

looks like a folded protein. In order to speed up this process, we will introduce a knowledge-based component to the search process.

### 3.2 Knowledge-Based Sampling

The next predictor will use a library of polypeptide conformations, taken from published structures in the PDB, to inform its random conformational sampling. In particular, at each step, the algorithm will choose an $n$-mer in the current structure and replace the backbone geometry with the backbone geometry of a corresponding $n$-mer from the library. An $n$-mer is a sequence of $n$ amino acids.

**Question 10:**
`sampleMove()` *in* `FragmentPredictor` *has been implemented for you. This function is simple and involves a call to PyRosetta's* `ClassicFragmentMover` *object.*

*(a) Run the* `FragmentPredictor` *for 1000 iterations on* `hras.pdb` *and watch it search the conformational space in PyMOL. Do this for the set of 9-mers and 3-mers.*

```
python3.8 predict.py pdbs/sequence.pdb frag9.pdb -frag9 -pymol
-1000
python3.8 predict.py pdbs/sequence.pdb frag3.pdb -frag3 -pymol
-1000
```

*Include a screen-shot of the structure and of the Ramachandran plot after 1000 iterations.*

*(b) As before, now run each fragment predictor for 100,000 iterations. Observe the final structure in PyMOL and view the Ramachandran plots of the predicted structures. Include a screenshot of the structure and Ramachandran plot.*

*(c) Describe your predictions after 1000 and after 100,000 iterations. Can you distinguish any distinctive secondary structures in PyMOL or in the Ramachandran plot? How do both of these predictions compare to each other and to the dihedral prediction?*

**Question 11:** *Let's compare the convergence and computational properties of the dihedral and fragment predictors.*

*(a) Which method (dihedral or fragment) seems to converge to a reasonable structure more quickly? Why is this the case?*

*(b) Which method (dihedral or fragment) is less computationally expensive for each step of the prediction (i.e. each single update in conformation)? Explain why this is the case.*

It is also possible to combine both dihedral and fragment based methods. Try running your predictors in succession using the following commands. We generally start with coarse-grained knowledge-based methods, and then move on to more refined sampling. The output of these commands will be saved in **out/result.pdb** and will be required for the following questions.

```
python3.8 predict.py pdbs/sequence.pdb frag9-100K.pdb -frag9
-100000
python3.8 predict.py out/frag9-100K.pdb frags.pdb -frag3 -100000
python3.8 predict.py out/frags.pdb result.pdb -dihedral -100000
```

### 3.3 Side-Chain Packing

So far, the structure prediction algorithms we've used have focused on sampling conformations of the protein backbone. The backbone is crucial in determining secondary and tertiary structure, but it is not the whole story.

As PyRosetta modifies a protein's conformation according to the coarse-grained Rosetta energy function, it does not model the full geometry of side-chains. When the structure is converted back to an all-atom representation, the side-chains can take on highly improbable conformations.

To measure the quality or "closeness" of our prediction, we will use PyMOL to calculate the **root mean square (RMS)** distance between the atoms of the known structure and the corresponding atoms of the predicted structure. Here RMS is calculated using the 3D coordinates of each atom, after the structures have gone through an alignment step.

**Question 12:** *In PyMOL, load in* `helix.pdb` *and* `bstrand.pdb` *from the "pdbs" folder, and load your final predicted structure* `result.pdb` *from the "out" folder. We will compare how well we did in predicting the alpha helix and beta strand portions of our protein.*

```
PyMOL> select result-helix, (result and resi 152-168)
PyMOL> align result-helix, helix, quiet=0
PyMOL> center helix
```

```
PyMOL> select result-bstrand, (result and resi 37-58)
PyMOL> align result-bstrand, bstrand, quiet=0
PyMOL> center bstrand
```

*What is the RMS distance between each pair of predicted and crystal structures?*

***Note**: It is okay if secondary structures are not present in the aligned region. However, if there are no secondary structures in your entire predicted structure, you likely have a bug.*

To predict the side-chain orientations more realistically, PyRosetta provides methods which perform similar Monte Carlo sampling on the side-chains. These sampling techniques can also be used in protein design, but here, we will just use them to predict the existing side-chains' orientations.

```
python3.8 pack.py <input>.pdb <output>.pdb
```

**Question 13:** *Look at the side-chains for the protein after packing. What are the RMS distances now for the alpha helix and beta strand sections? Is the relative magnitude of change in RMS distance after packing roughly the same or fairly different between the alpha helix and beta strand? Briefly explain why this is the case.*

**Question 14:** *Compare the final structure (after side-chain packing) to the ground-truth crystal structure (*`hras.pdb`*). Load in hras from the "pdbs" folder and perform the align command. Report the RMS distance and include a Ramachandran plot.*

**Question 15:** *How well did your algorithm predict the secondary structures of hras? Which were predicted more accurately, alpha helices or beta strands? Based on the types of predictions we made (dihedral- and fragment- based), why do you think this is? What are some of the pros and cons to each methodology (dihedral- vs. fragment- based prediction)?*

# 4 Feedback

Again, we'd love some feedback on the assignment! You will receive full credit for this portion for providing any response. We encourage constructive criticism.

**Question 16:** *What was your favorite aspect of the assignment? What was your least favorite aspect of the assignment? Why? Any suggestions for improvement?*

**Question 17:** *Approximately how long did this assignment take you? Where did you spend most of this time?*

# 5 Submission Instructions

As before, complete your writeup in the text editor of your choice, and then convert to a pdf. Upload your writeup and your implementation of `predictor.py` to gradescope.

If you have any issues submitting, let us know!