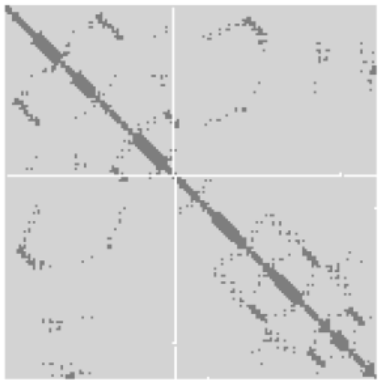
Homework #2: Building a protein model guided by a contact map.



In this homework you are going to write a PyRosetta script that will take as input a sparse 2D contact map:

The output of this will be a three-dimensional model that satisfies the constraints *and* is favorable by Rosetta's energy function.

Input format:
The constraint file and sequence will be provided. The format of the constraint file:

```
  1 17
  1 19
  …
  78 89
  79 89
```

Where each line shows a predicted contact (that is, Cbeta – Cbeta distances within 6Å).

The format of the fasta file is:

```
  > name
  ATSCYNPIQE…
```

Where the second line provides the amino-acid sequence (one-letter code) of the target protein.

I have provided a .tgz file containing 8 fasta files and 8 corresponding constraint files (extract with '`tar xvfz hw2.tgz`') For all 8, I have also provided a PDB file of the native conformation. The remaining assignment will focus on these 8 examples.

Part 1. Naïve algorithm

We are going to approach this problem naively at first. We are going to write a simple protocol that:

1) Sets constraints on the pose from the constraint file
2) Sets the backbone torsion angles to random values
3) Relaxes the structure with constraints

A few PyRosetta code snippets are listed that might aid in this process:

1. To set constraints on the target pose between the Cbeta of residue *i* and residue *j*:

```
id_i = AtomID(pose.residue(i).atom_index("CB"),i)
id_j = AtomID(pose.residue(j).atom_index("CB"),j)
ijfunc = rosetta.core.scoring.constraints.BoundFunc(
     0.0, 6.0, 1.0, 'cst1');
cst_ij = rosetta.core.scoring.constraints.AtomPairConstraint(
     id_i, id_j, ijfunc)
pose.add_constraint(cst_ij)
```

2. To set the scorefunction to use constraints:

```
scorefxn =
rosetta.core.scoring.ScoreFunctionFactory.create_score_function(
    "ref2015_cst.wts")
```
Or
```
scorefxn = get_fa_scorefxn()
scorefxn.set_weight(
    rosetta.core.scoring.atom_pair_constraint, 1.0)
```

3. To relax a pose:
```
fastrelax = rosetta.protocols.relax.FastRelax(scorefxn, 5)
fastrelax.apply(pose)
```

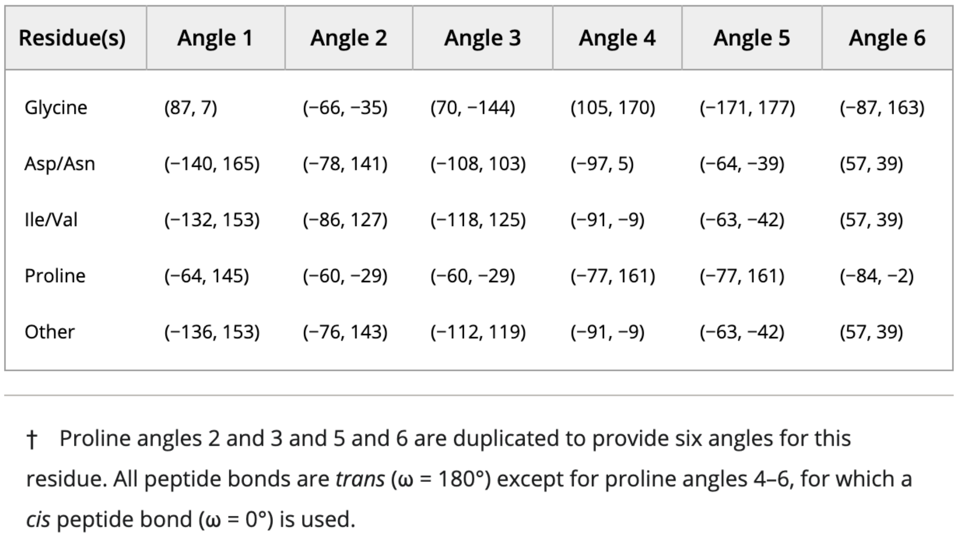
Choose **four** of the eight targets (your choice) and run your method. Since you are randomly setting backbone torsions, each time you run it, it will yield different results. Try running it several different times (at least 10, and more if your computer is fast enough), and save the structure and score of the three lowest-scoring:

- (Using PyMol) How do the three structures compare to each other? Does this suggest sampling has converged? How do these structures compare to the native?
- Add constraints and run an identical relax on the native. How do the energies compare between the sampled decoys and the relaxed natives? Look at energies: a) of the constraints only, b) of the Rosetta scoreterms only, and c) the sum of both.

Part II: improving the protocol.

In this part, we will try to improve the sampling method from part I. There are two things we will try to improve: a) the backbone initialization, and b) the minimization strategy

Improved backbone initialization. Knowing what we know about the Ramachandran distribution, setting phi/psi angles randomly is not an ideal strategy. In Part 2, we are going to sample from the Ramachandran distribution. A 2001 paper (https://www.ncbi.nlm.nih.gov/pubmed/11276088) has the following table:



Residue(s)	Angle 1	Angle 2	Angle 3	Angle 4	Angle 5	Angle 6
Glycine	(87, 7)	(−66, −35)	(70, −144)	(105, 170)	(−171, 177)	(−87, 163)
Asp/Asn	(−140, 165)	(−78, 141)	(−108, 103)	(−97, 5)	(−64, −39)	(57, 39)
Ile/Val	(−132, 153)	(−86, 127)	(−118, 125)	(−91, −9)	(−63, −42)	(57, 39)
Proline	(−64, 145)	(−60, −29)	(−60, −29)	(−77, 161)	(−77, 161)	(−84, −2)
Other	(−136, 153)	(−76, 143)	(−112, 119)	(−91, −9)	(−63, −42)	(57, 39)

† Proline angles 2 and 3 and 5 and 6 are duplicated to provide six angles for this residue. All peptide bonds are *trans* (ω = 180°) except for proline angles 4–6, for which a *cis* peptide bond (ω = 0°) is used.

Use this table to provide your initial random values for phi and psi.

Improved minimization. To avoid getting stuck in local minima, it may be beneficial to try minimizing not with all constraints enabled right away, but instead:

- first minimizing with constraints close in the polypeptide chain (those with short *sequence separation*)
- then adding constraints with longer sequence separation
- and so on.

So, one might first add constraints between residues separated by 5 residues or less, then adding on those separated by 10, then 15, etc.

For this step I recommend using a "shorter" all-atom relax (since you will be calling it more than once in each trajectory):

```
    fastrelax = rosetta.protocols.relax.FastRelax(scorefxn, 1)
```

The second argument (here 1) defines the number of repeats of refinement to run.

Run this improved protocol on at leas **four** of the eight cases, again with at least 10 models each, and more if possible. How are you doing? As in part 1, compare models to the native both subjectively (in PyMol) and comparing energies.

Part III: add secondary structure prediction

Now try adding secondary structure prediction on top of this approach. Use the secondary structure prediction tool *psipred* (http://bioinf.cs.ucl.ac.uk/psipred/) to predict the secondary structure of each protein. Use this to guide phi/psi sampling (recall phi/psi = -60/-50 in helices and -140/+130 in strands) in predicted secondary structures.

There are a few other advanced things you can try:

- Change the minimization strategy: minimization with a subset of constraints, changing constraint weights, minimization with a subset of DOFs, alternating Cartesian and internal-coordinate minimization
- Monte Carlo sampling: alternate perturbing the structure, minimizing, and accepting or rejecting the change.
- Large-scale sampling (sample 100 random structures or more)
- Something else

Anything goes here, feel free to try what you want! Try listing at least one improvements you tried and how they worked (or did not work). Ideally, you are able to get near-native conformations (4A or lower RMS) for at least two structures.

What to hand in: Your code and a (at most) 1-page writeup of the methods.