CS483 Assignment #3:

Working with Angles and Distances

Due date: Thursday Feb. 26. **"Early Bird" Due date: Tues. Feb. 24 to get 5 bonus marks.**

Marks: (The assignment will be marked out of 80).
Ex. 1: [Part(a)10 + Part(b)6 = 16], **Ex. 2**: [12], **Ex. 3**: [12], **Ex. 4**: [16] , **Ex. 5**: [Part (a) 16 + Part (b) 12 = 28]

Main objectives of this assignment:

- Do computations involving bond angles and interatomic distances
- Get more experience with the collection of statistics and generation of plots.
- Apply linear algebra in the development of an algorithm
- Get experience in the generation of a visual representation related to molecular events.

Exercise 1: Four Ramachandran Plots

1(a)

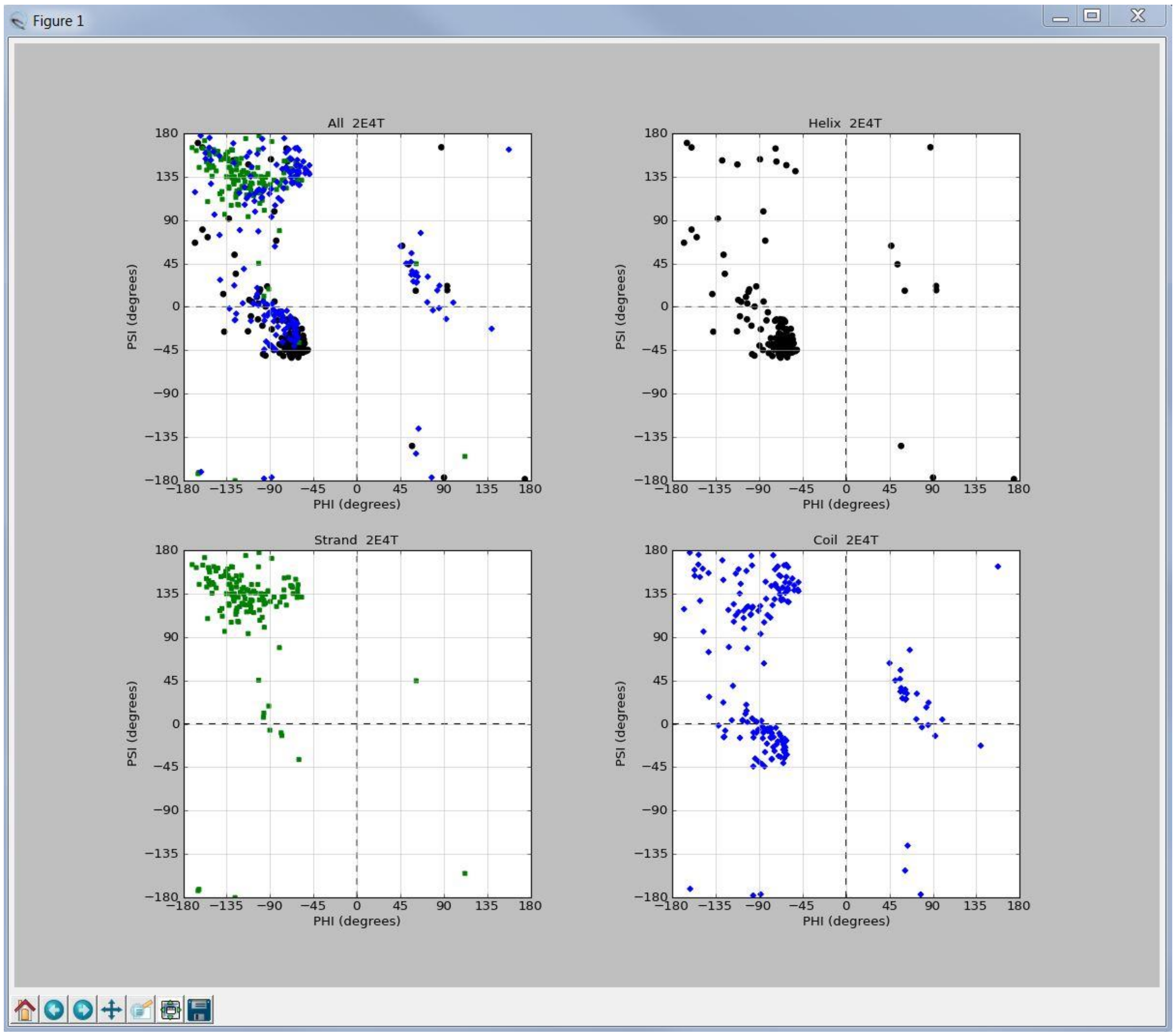
For this exercise, start by reading Section D.5 of the appendix in the text. It gives code that can be used to generate a 2D scatter plot. This would be very useful in the generation of a Ramachandran plot, for example. Unfortunately, the usual Ramachandran plot has a limitation. Even though there is a tendency for (phi, psi) pairs to cluster in different areas of the plot, it is often difficult to separate points associated with different secondary structures (Helix, Strand, and Coil) because in areas of high point density we will get a lot of marker overlap. For example, it is often difficult to know which points correspond to coil secondary structure since very short segments of a coil can have phi, psi angles that are consistent with helix or strand. In this exercise we wish to generate a figure with four plots arranged in a 2 by 2 array. The top leftmost plot will be the standard Ramachandran plot, the top rightmost plot will be for phi, psi pairs found in helix secondary structures, the bottom leftmost plot will be for strand secondary structures, and the bottom rightmost plot will be for coils. To do the plots you should modify the code shown on page 370 of the text. As a hint to get you started, you can replace the statement:

```
ax = fig.add_subplot(111, aspect = ‘equal’)
```

with the statements:

```
plotIx = 0
for ssCode in ("All", "Helix", "Strand", "Coil"):
    plotIx += 1
    ax_D[ssCode] = fig.add_subplot(2,2,plotIx, aspect = 'equal')
```

Other statements will be required to define the attributes associated with the entries in the `ax_D` dictionary. Note that the first three parameter in the `add_subplot` invocation defines the dimensions of the plot display (2 by 2) and also the position of a particular plot within that display of subplots. The mainline of your script should ask the user for a PDB identifier. Subsequent code will scan the standard residues, extract the phi, psi dihedral angles and then use them to define points in the four scatter plots. The next figure shows the appearance of the subplots. Note that different markers have been used for the points corresponding to different secondary structures: black circles for helix, green squares for strand, and blue diamonds for coils.



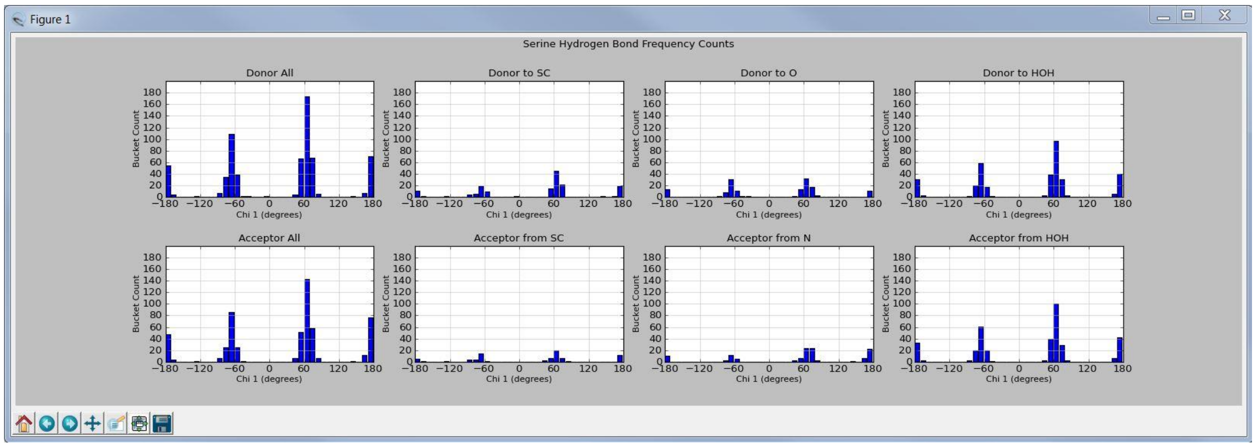
1(b) Modify the script in part (a) so that it works with chi1 and chi2 dihedral angles instead of phi and psi. Your script should skip any residue that has only one chi dihedral (or none as is the case for GLY and ALA). You should notice that there is a tendency to have 9 clusters in these scatter plots because the chi1 dihedral has 3 rotameric preferences and for each of these the chi2 dihedral has 3 rotameric preferences. This clustering would be more pronounced if we accumulated data over several protein files.

Exercise 2: Serine Chi1 values and hydrogen bonds

Motivtion: Recall the BBDep column of the accuracy table at: http://dunbrack.fccc.edu/scwrl4/. The BBDep column indicates that if you do not use a chain packing algorithm and simply use the most probable rotamer the accuracy of the chi1 dihedral (when compared to the crystal conformation) will be reduced. This is especially true for serine even though it would seem to be more predictable since it has only one dihedral angle. To get a more informed perspective on this issue, we will generate some plots to see how the hydrogen bonding of the side chain is influenced by its immediate environment.
For this exercise you are to generate a figure with 8 plots. Each plot will be a bar chart with positions for 36 bars. The x-axis spans the interval [-180, +180] and the y-axis spans the interval [0, 200]. The width of a bar will correspond to a subinterval of 10 degrees on the x-axis and each bar will have a height that specifies the number of times a chi1 angle is in that subinterval. The eight plots are to be organized as an array with 2 rows and 4 columns. The top row deals with serine acting as a hydrogen bond donor and the bottom row deals with serine acting as a hydrogen bond acceptor.

The first subplot of the top row is a bar chart showing the frequency distribution of serine chi1 angles without any constraint imposed on the serine residue other than it being a donor. The next three subplots place restrictions on the serine donor. The second subplot is for serine donors that have an acceptor in a standard residue sidechain (i.e. the acceptor is not an oxygen in the protein's backbone). The third subplot is for serine donors with a backbone oxygen as the acceptor and the last subplot is for serine donors that use water as an acceptor.

The first subplot of the bottom row is a bar chart showing the frequency distribution of serine chi1 angles without any constraint imposed on the serine residue other than it being an acceptor. The next three subplots place restrictions: The second subplot is for serine acceptors that have a donor in a standard residue sidechain (i.e. the donor is not a nitrogen atom in the protein's backbone). The third subplot is for serine acceptors with a backbone nitrogen acting as the donor and the last subplot is for serine acceptors that use water as a donor.
To get data for the script you should access protein chains that are specified by the cullpdb_pc20_res1.6_R0.25_d141204_chains2395.gz list that can be found at: http://dunbrack.fccc.edu/Guoli/pisces_download.php. It is not necessary to use all the chains in the list. Just keep accessing files until you have processed 500 serine residues. My results:



Exercise 3: Atomic distances and Hydrogen bonding

Motivation: Researchers[1] have discovered that hydrogen bonds typically exist over a somewhat restricted distance. This exercise is meant to give you some perspectives on these distances.

Working with the data set used in the previous exercise, design and implement a script that answers the following questions dealing with hydrogen bonds involving the serine OG atom in both the donor and acceptor roles. The answers for all questions should be based on at least 100 instances of the scenario described and for each scenario you should report the average distance, minimum distance observed, and maximum distance observed. These three distances should be reported in the output of the script. In a separate discussion to be added to your submission you should compare these values with the sum of the van der Waals radii of the two atoms involved. Note that Chimera has more than one specification for an atom radius: https://www.rbvi.ucsf.edu/chimera/docs/UsersGuide/midas/vdwrad.html.

a) Serine OG acting as a donor and backbone oxygen acting as the acceptor.

b) Serine OG acting as a donor and any other acceptor that is not backbone oxygen or oxygen in water.

c) Serine OG acting as an acceptor with backbone nitrogen acting as the donor.

d) Serine OG acting as an acceptor with any other donor that is not backbone nitrogen or oxygen in water.

Exercise 4: Serine and steric collisions

Motivation: Recall the sidechain packing problem; we know the positions of all atoms in the backbone of a protein and we know the residue type rooted at each alpha carbon. How do we select an appropriate rotamer for each side chain so as to avoid steric collision and, moreover, attain the minimum potential energy (presumably achieved by the conformation seen in the crystal form of this protein)? The research paper by Lovell et al.[2] reports that the mode (most common setting) for the serine rotamers are defined by the Chi1 angle having values 62, -177, and -65 degrees. These values are designated as the **p**, **t**, and **m** settings for the Chi1 angle. Note that your plots from Exercise 2 should show a consistency with these values.

Early versions of sidechain packing algorithms typically dealt with avoidance of steric clashes and ignored any energy reduction provided by the formation of hydrogen bonds. The objective of this exercise is to gather more data related to steric collisions and the rotameric settings of serine. Roughly speaking, we want to evaluate the amount of "wiggle room" for serine. Expressed more precisely: Does the Chi1 setting for serine (as seen in a PDB file) typically put the OG atom very close to another atom (thus lowering the pairwise potential energy defined by the Lennard-Jones potential) or does the PDB Chi1 setting result in the OG atom being well away from any nearby atom.

We will adopt the following terminology:

Proximal distance will be the distance between the OG atom of the serine residue and the *nearest* atom in a neighbouring standard residue when Chi1 has one of the settings that is not the one being used by the

[1] R.W.W. Hooft, C. Sander, and G. Vriend. Positioning hydrogen atoms by optimizing hydrogen-bond networks in protein structures. *Proteins*, **26** (1996) 363-376.
[2] S.C. Lovell, J.M. Word, J.S. Richardson, and D.C. Richardson. The penultimate rotamer library. *Proteins*, **40** (2000) 389-408.

conformation in the PDB file. For example, if the serine residue had a PDB setting for Chi1 that was 60 degrees then this "wild" setting would be closest to the p mode and we would want to examine two alternate settings, namely: -177 and then -65 degrees (evaluating the proximal distance for each setting).

Wild distance will be the distance between the OG atom of the serine residue and the nearest atom in a neighbouring standard residue when Chi1 has the setting found in the conformation seen in the PDB file.

Your script should process 500 serine residues using the same chains as those used in Exercise 2. For each serine residue do the following:

Determine the rotameric setting (p, t, or m) for the serine residue. In other words, determine the angle in the set of modes (62, -177, -65) that is closest to Chi1. Calculate the wild distance. Change Chi1 so that it takes on both of the other mode settings. For each setting we calculate the proximal distance. There are three outcomes:
i) The proximal distance is less than the clash threshold (say, 2.5) and this indicates a steric collision,
ii) The proximal distance is less than the wild distance.
iii) The proximal distance is greater than the wild distance.

Since we are looking at two additional Chi1 settings for each of the 500 serine residues we have 1000 new Chi1 settings to evaluate and they will be distributed across these three outcomes. Your script should evaluate and print out the number of occurrences for each of the three types of outcome.

Why are we interested in these numbers? Naïve side chain packing algorithms tend to choose rotamer settings that are consistent with the last outcome since that rotamer is considered to be more likely to avoid steric collision (nearest atom at the largest distance from OG). However, in these case it would seem that the wild type conformation has chosen a Chi1 setting such that OG is closer to its nearest neighbour.

Hints and cautions:

- Read about the `Shell` class that can be found in the StructBio download. You can use two instantiations of the class: One to determine if there is a steric clash and another to find the nearest neighbour atom (set `maxR` to 2.5 for clash discovery and set `maxR` to 5.0 when you need to rapidly scan a short list of atoms to get the nearest neighbour). (With the help of the Shell class, I did the entire script in less than 110 lines of Python).
- Be sure to work with serine residues in the specified chain and not the entire protein. It is acceptable to have a situation where the neighbouring atom of OG is in a different chain.
- Skip SER residues that have alternate locations (seen in the Chimera display as two side chains sprouting from the same alpha carbon). An easy test for the presence of alternates: `len(r.altLocs) != 0`
- Here is the output that I got when processing 50 residues:

```
Collision Threshold:                2.50
Total number of SER residues :        50
Number of Chi1 giving a collision:    11
Number of chi1 with proximal distance
less than wild distance:              34
Number of chi1 with proximal distance
greater than wild distance:           55
```

Exercise 5: An urchin plot for serine hydrogen bonds to backbone atoms

5(a) **Getting serine hydrogen bond donors and acceptors into a common frame of reference:**
Start by writing a function with the following header and explanatory comments:

```
def coordsInLocalFrame(originAtm, zDirectionAtm, xProjDirAtm, target):
    # All arguments are atom objects.
    # This function will return the array coordinates of target
    # in a frame of reference established as follows:
    # The coordinates of the first atom argument define the origin.
    # The next argument defines the direction of a unit normal for the z axis.
    # The function now has enough information to define a plane going through
    # the origin and normal to the z-axis.
    # The x-axis will be defined by a unit normal in the plane with a direction
    # defined by the projection of the xProjDirAtm coordinates onto the plane.
    # The y-axis will be the normalized cross product of the two normals just defined.
```

As discussed in class you will be invoking this function with `originAtm` set to a serine alpha carbon, `zDirectionAtm` set to the serine beta carbon, `xProjDirAtm` set to the serine nitrogen, and some other atom (a backbone oxygen or nitrogen) will be the `target`. Work with the function to do the following: Using the same data set as in Exercise 2, go through enough residues to get at least 200 instances where the OG atom of SER is acting as a donor to the O atom in the backbone and at least 200 instances where the OG atom of SER is acting as an acceptor of the N atom in the backbone. For each SER residue satisfying at least one of these requirements call the function with arguments as described above and write a line out to a file that will retain the coordinate data. An output line should start with an "`hBondCode`" that specifies whether the OG atom is acting as a donor ("D") or as an acceptor ("A"). The code will be followed by six floating point numbers representing the coordinates of the OG atom and its hydrogen bond partner both expressed in the local frame of reference established by the `coordsInLocalFrame` function. Note that you will also need to call your function with the OG atom acting as the target so that you can get the first set of coordinates in an output line. Here are the first three lines produced by my script:

```
D     -1.294  -0.044   2.097        -2.732  -1.505  -0.014
D      0.548  -1.187   2.058         3.517  -0.348   1.427
D      0.457  -1.285   1.962         3.064  -1.623   1.238
```

Notes: As in the previous exercise, process only the SER residues in the specified chain and skip any SER residue that has alternate locations. Also, be careful that the O acceptor for a donor OG is in the backbone and not the oxygen atom of a water molecule.

5(b) **Showing hydrogen bonds in an urchin plot:**
Write a script that will read the data file produced in the previous exercise. Start with an empty display and use the `Solids` class in the StructBio download to generate a visual representation of an XYZ coordinate system using the `Solids` class:

```
axes = Solids("axes")
axes.addArrow(array((0., 0., 0.)), array((5., 0., 0.)), .05, colorByName("red"))
axes.addArrow(array((0., 0., 0.)), array((0., 5., 0.)), .05, colorByName("green"))
axes.addArrow(array((0., 0., 0.)), array((0., 0., 5.)), .05, colorByName("blue"))
axes.display()
```

Note that the axes can be distinguished by their RGB coloration. The call to `colorbyName` is for a utility function that is also in the StructBio package. If you want to avoid the calls, use the tuple (1., 0., 0.) for red, (0., 1., 0.) for green, and (0., 0., 1.) for blue.

Your script should read the data file and set up small spheres, one for each donor and acceptor set of coordinates. All spheres should have a radius of 0.05 with a red colour if it represents an acceptor and a blue colour if it represents a donor. You should work with four instantiations of the `Solids` class, one for each category of atom: OG donor, OG acceptor, N donor, and O acceptor. Later, after the script has completed execution, the separate instantiations will allow you to selectively hide or display these sphere categories by working with Chimera's Model Panel. Use the `Lines` class to draw lines between the hydrogen bond donors and their acceptor partners. A line should have the same color as its donor atom. The final scene should look like the display shown in the next figure. Note the apparent clustering of the donors and acceptors in the 3D space.