Graduate Topics in Biophysical Chemistry – CH 8990 03
Assignment 0 (Programming Assignment)

Due Monday, March 19

Introduction and Goals

It is virtually impossible to be a successful scientist today without some understanding of computer programming. Modern instrumentation relies on computation to process and store large datasets, and frequently it is necessary to use programming as a means of examining the data in novel ways. This is especially true in the field of protein science because of how protein coordinates are stored. Complex molecular structures require proportionately complex methods for data analysis, and even those who aren't involved in molecular simulation must have some practical understanding of how to manipulate structural data.

If you already know a programming language, you should strongly consider using that language to complete the assignment. I know many programming languages, including C, C++, Fortran, Python, and Perl, and I can offer some assistance if you get stuck. If you don't already know a programming language, this assignment will force you to learn one. I don't care which programming language you use, but I would strongly recommend using Python if you haven't programmed before. It's straightforward and easy to use while being very powerful. After completing the on-line tutorial, you will have the tools to write very sophisticated programs, and I know of no other language where the startup time is so small. The tutorial for Python 2.7 is found at http://docs.python.org/tutorial/.

Another goal of this assignment It will also force you to understand how structural data is stored in the PDB and how to access that data. PDB files contain all the information needed to reconstruct the structure of a protein or nucleic acid: there are atom names, residue names, residue numbers, and the X, Y, Z positions for each atom. With the exception of some "header" information, PDB files are simply a list of this information, which each line corresponding to a separate atom. Your program will have to make sense of this file and perform some useful computation on the coordinates.

Although not explicitly required, a final goal of this assignment is familiarize you with the UNIX operating system environment. The Python interpreter is available as a download for PCs, and all Macs already have Python installed in their UNIX back-end, but as a student of this class, you will be given access to my Linux server so you can write your program there if desired. If you would like an account, please email me and I will give you the details in class. You will not be required to use my Linux server, but I guarantee that knowing UNIX/Linux will benefit you in the future. Employers like to see UNIX experience on CVs, but it also frequently comes up in scientific environments. A good UNIX tutorial can be found at http://www.ee.surrey.ac.uk/Teaching/Unix/.

There is a lot to learn from this assignment, and very little of this material will be taught in class. Therefore, **if you do not have programming background or experience with UNIX, I recommend you start this assignment *right now***. Even though you have more than two

months to complete this assignment, that time will pass quickly. I am happy to offer help and advice, but waiting until Spring Break will probably be too late to learn everything needed for this assignment.

The Project

The project is (conceptually) simple: You are to write a program that computes the Ramachandran plot coordinates for a given protein. The protein you will use is Staphylococcal nuclease (Snase), and you will examine two PDB files: PDB ID 2SNS and 1SNC. Your program must output the φ, ψ coordinate pairs for each residue (one per line). You must then submit a plot of those coordinates. You may create your plot in any program that you like (e.g. Microsoft Excel); however, a 10% bonus will be given if the plot is created in the UNIX program Grace (xmgrace).

As you think about your code, you will need to consider a structured approach to solving the problem. The following points may be helpful:

1. An approach to solving the problem would be to scan through the PDB file and collect the coordinates of the necessary atoms as you go along. Then, when you have the atoms, you can calculate φ and ψ and display them to the screen. Since you're not interested in anything other than the backbone, it's safe to ignore all other atoms. The atoms that define φ and ψ are:

   φ: C'_{i-1}- N_i- $C\alpha_i$- C'_i
   ψ: N_i- $C\alpha_i$- C'_i- N_{i+1}

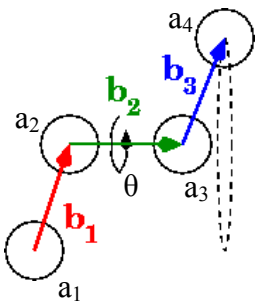
2. PDB files have a standard text format, the definition for which can be found at http://deposit.rcsb.org/adit/docs/pdb_atom_format.html. You are most interested in the ATOM records, and you can safely ignore the other lines for this project.

3. The first and last residues in the PDB file do not have a φ, ψ pair, because they are missing one of the necessary atoms.

An example Ramachandran plot for 2SNS is included at the end of this assignment. Your solution needn't be sophisticated, but it must nevertheless have axis labels and look professional.

Calculating Dihedral Angles

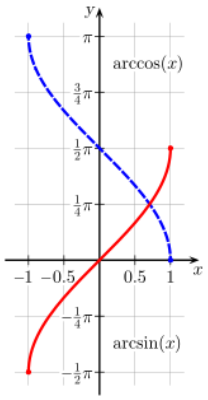
Calculating dihedral (torsion) angles is not as straightforward as you might think. The main goal of this assignment is to teach you how to write a program that computes useful information. While vector algebra is an important part of this assignment, it's not intended to be an obstacle. An expression for computing the dihedral angle is given succinctly in Wikipedia, at http://en.wikipedia.org/wiki/Dihedral_angle. If four atoms, a_1, a_2, a_3, and a_4 form a dihedral angle, let the vectors $\vec{b}_1$, $\vec{b}_2$, and $\vec{b}_3$ connect those atoms, each vector pointing from the N-terminus to the C-terminus of the



chain (as shown to the left). Then, the dihedral angle for the bond between a_2 and a_3 is given by (eq. 1):

$$\theta = \text{atan2}(|\vec{b}_2|\vec{b}_1 \cdot (\vec{b}_2 \times \vec{b}_3), (\vec{b}_1 \times \vec{b}_2) \cdot (\vec{b}_2 \times \vec{b}_3)) \quad (1)$$

The function atan2 gives us the correct angle over the correct range (-π to π). A simpler approach would use the inverse trigonometric functions, but this quickly leads to problems. As shown to the right, both $\sin^{-1}$ and $\cos^{-1}$ have a limited range, and neither by itself can tell you the correct angle given a value between -1 and 1. On the other hand, the normal $\tan^{-1}$ function, which takes the ratio of sine and cosine, is also limited: For example, if the sign of both sine and cosine are negative, the ratio will be positive (since the ratio of any two negative numbers is positive). Thus, the standard $\sin^{-1}$, $\cos^{-1}$, and $\tan^{-1}$ functions by themselves are *not sufficient* to calculate an angle uniquely on the complete range from -π to π.



The atan2 function takes the sine and cosine of an angle as separate arguments. The first argument is the sine of an angle, and the second argument is the cosine. Looking at these two values individually, atan2 is able to avoid the sign cancellation problems that plague $\tan^{-1}$. It *can* determine an angle on the complete range from -π to π. It does this by examining the sign of the sine and cosine arguments and by using some additional logic to arrive at the right answer.

An example demonstrating how to use the atan2 function in python is provided for you on the course web page. The syntax of python is very easy to understand, so even if you are working in another language you should be able to grasp the logic. Note that for every vector used in the program, three numbers are needed: one to store each of the x, y, and z components of the vector.

How is eq. 1 derived? The first insight is that the atoms a_1, a_2, and a_3 form a plane. So do atoms a_2, a_3, and a_4. Recall that the normal vector is the vector that is perpendicular to a plane. If we knew the normal vectors to the first and second planes, we could find the dihedral angle as the angle between the normal vectors. We can calculate the normal vectors using the cross products between $\vec{b}_1$, $\vec{b}_2$, and $\vec{b}_3$. Specifically, $\vec{b}_1 \times \vec{b}_2$ will yield a vector that is normal to the first plane, and $\vec{b}_2 \times \vec{b}_3$ results in a vector normal to the second plane. For convenience, let's call these normal vectors $\vec{n}_1$ and $\vec{n}_2$.

Now our goal is to find the angle between the normal vectors $\vec{n}_1$ and $\vec{n}_2$. Given the discussion above, we want to find the cosine and the sine of that angle so we can use atan2 to find the correct value. Remember that the dot and cross products can tell us about the sine and cosine of an angle:

$$|\vec{n}_1||\vec{n}_2| \cos\theta = \vec{n}_1 \cdot \vec{n}_2 = (\vec{b}_1 \times \vec{b}_2) \cdot (\vec{b}_2 \times \vec{b}_3) \quad (2)$$

$$|\vec{n}_1||\vec{n}_2| \sin\theta = |\vec{n}_1 \times \vec{n}_2| = |(\vec{b}_1 \times \vec{b}_2) \times (\vec{b}_2 \times \vec{b}_3)| \quad (3)$$

The first expression, using the dot product, is straightforward, because the dot product already returns a (signed) scalar value. The sign of $\vec{n}_1 \cdot \vec{n}_2$ will reflect the sign of $\cos\theta$. Thus, you can see that the cosine argument in eq. 1 follows directly from eq. 2.

The second expression is a bit trickier, because we must not lose the sign information when we take the absolute value of the cross product in eq. 3. That is, we need an expression for $|\vec{n}_1 \times \vec{n}_2|$ that can determine whether we have $\sin\theta$ or $\sin(-\theta)$. Otherwise, the value used in atan2 will always be positive, and we still won't be able to calculate the correct dihedral angle. This problem is solved with a clever trick to reduce the right hand side of eq. 3 to a *scalar triple product* (*e.g.*, $\vec{b}_1 \cdot (\vec{b}_2 \times \vec{b}_3)$):

$$\begin{aligned}
\vec{n}_1 \times \vec{n}_2 &= (\vec{b}_1 \times \vec{b}_2) \times (\vec{b}_2 \times \vec{b}_3) \\
&= -(\vec{b}_2 \times \vec{b}_1) \times (\vec{b}_2 \times \vec{b}_3) \\
&= -[\vec{b}_2 \cdot (\vec{b}_1 \times \vec{b}_3)]\vec{b}_2 \\
&= [\vec{b}_2 \cdot (\vec{b}_3 \times \vec{b}_1)]\vec{b}_2 \\
&= [\vec{b}_1 \cdot (\vec{b}_2 \times \vec{b}_3)]\vec{b}_2 \\
|\vec{n}_1||\vec{n}_2| \sin\theta &= [\vec{b}_1 \cdot (\vec{b}_2 \times \vec{b}_3)]|\vec{b}_2|
\end{aligned}$$

The second step and the next-to-last step follow from the properties of the scalar triple product. The cross product of the normal vectors should point along the a_2-a_3 bond (i.e., $\vec{b}_2$), and it will be positive or negative depending on the sign of the dihedral angle. With this approach, we can maintain the sign information for use with the atan2 function, and the resulting angle will be on the range $(-\pi, \pi)$.

What to Submit

When you have finished your program, submit the following, either digitally or in printed form, by the beginning of class on March 19th:

1. A complete copy of your code.

2. Your Ramachandran plots for 2SNS and 1SNC.

3. Your xmg data files (if you used Grace to create your graph). This should be sent digitally.

4. A brief assessment describing which structure (either 2SNS or 1SNC) is more accurate, along with an explanation why. (No more than two or three sentences are needed here.)

Example Solution (2SNS)