COMP 116
Programming Assignment #6:
File I/O and the Protein Data Bank
Due Date: Sunday, November 21, 2010

Introduction

A protein is a molecule that is a sequence of amino acid residues. The Protein Data Bank records the 3D structures known for protein molecules. Two examples are HIV Protease (7hvp), an important AIDS drug target, and green fluorescent protein (1gfl), which earned its discoverers the Nobel Prize in 2007. Many types of information are stored in a `.pdb` file; we will be interested only in lines that start with `'ATOM'`, and only in certain columns of these lines. The format of the `.pdb` file is described on the last page.

We will be working with the two proteins above, 7hvp and 1gfl. You can find the `.pdb` files describing them in the assignment resources folder on Sakai.

In this assignment we stress that you use good programming style — this includes commenting your code, using meaningful names for your variables, and if you're asked to implement a specific function, *please* use the same name we have supplied. Points *will* be deducted for unclear code.

Tasks

1. Write a function `readPDBFile('filename')` that will read the atoms for a protein stored in a `.pdb` file whose name is specified in quotes. The first line of your function should be:

   ```
   function [anum, aname, resno, coords] = readPDBFile(infile)
   ```

   The output variables should be set as follows:

   anum column vector with the serial number for each atom

   aname $n \times 4$ string array with 4-letter atom name for each atom

   resno column with a residue sequence number for each atom

   coords $n \times 3$ matrix with (x,y,z) coordinates for each atom

   The string `aname` should be in upper-case.

2. Write a function `drawCA(aname, coords)` that uses the MATLAB function `plot3` to draw the **Ca backbone** of the protein; i.e. it should connect all atoms with the name `'CA'` in sequence, and ignore all other atoms.

3. Write a function `hBonds(anum, aname, resno, coords)` that looks for pairs with a Nitrogen atom (second letter of `aname` is `'N'`) and an Oxygen atom (second letter of `aname` is `'O'`) whose distance is between 2.6 and 3.2 angstroms, inclusive. Such a pair is deemed to form a

hydrogen bond if the residue sequence numbers of these atoms differ by at least 2. Return a list containing the pairs of atom numbers for hydrogen bonding pairs.

4. Turn in your three functions as well as a script that does the following:
   (a) Read in the file `7hvp.pdb`.
   (b) Draw the Ca backbone for 7hvp.
   (c) Display how many hydrogen bonds were found in 7hvp.
   (d) Read in the file `1gfl.pdb`.
   (e) Draw the Ca backbone for 1gfl.
   (f) Display how many hydrogen bonds were found in 1gfl.

Hints

Opening and Reading a File

A file such as `'7hvp.pdb'` (for HIV protease) must be opened and given a file ID before it can be read:

```
fid = fopen('7hvp.pdb');
```

Then, each time you say

```
line = fgetl(fid);
```

the variable `line` will contain the next line read from the file as a string (character array). If the first six characters of the line are `'ATOM  '` then that line has interesting information. The numbers and names all occupy fixed positions on a line, so you can easily extract them using indexing and slicing as you would an array or matrix, then convert them from strings to numbers (if necessary) in your reader function. If your file ID is called `fid`, you can check whether there is still data left in the file to be read by using the command `feof(fid)`. If the function returns true, then there is no more data left; if it returns false, there is still data left to be read.

So for example, if you wanted to go through and read in each line of a file called 'myfile.txt' and display the first 4 characters of each line, you would do it like so:

```
fid = fopen('myfile.txt');

while (~feof(fid))
   line = fgetl(fid);
   disp(line(1:4))
end
```

3D Plotting

Up until now we have only plotted 2D data, but MATLAB can easily plot 3D data as well. For example, if we had an $n \times 3$ matrix called `pos` where each row represented the (x, y, z) coordinate of a point, we could plot a line through all of these points with points marked as red asterisks using the following command:

```
plot3(pos(:,1), pos(:,2), pos(:,3), '-r*')
```

The line specifier works in precisely the same way as it did for 2D plotting.

Other Hints

- `upper()` and `lower()` can change the case of strings.
- `num2str()` and `str2num()` can convert a number into a string and a string into a number, respectively.
- To compare strings, use `strcmpi()` (which is case-insensitive). For example, `strcmpi('Yes', 'yes')` returns 1 (true) and `strcmpi('Yes', 'No')` returns 0 (false).
- The distance between two 3D points (x_1, y_1, z_1) and (x_2, y_2, z_2) is equal to:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

- For the hBonds function, you need to look at every possible pair of atoms. You also don't want to repeat pairs (for example, the pair (217,892) and the pair (892,217) should be considered the same). To handle this, think of the problem this way: you can start by looking at every single pair that includes atom #1. This means looking at #1 paired with every atom after it. After that, you don't need to ever look at atom #1 again. Then, you look at every single pair that includes atom #2 by starting with #2 and pairing it with every atom after it. You don't need to check the pair (#2,#1) because that pair got checked on the first run-through. Therefore, if I have an array called `A` and want to *iterate* through every single unique pair of elements of A, I can do it as follows:

```
for i = 1:length(A)-1
   for j = i:length(A)
      A(i)  % This is the first element of our pair
      A(j)  % This is the second element of our pair
   end
end
```

- Comment your code and give your variables meaningful names. This will not only help you keep your thoughts organized as you work, but will also help us assist you if you have questions or issues with your code.

Format of ATOM Records

Below is an example of a line in a `.pdb` file that corresponds to an `ATOM`. Above the line is an "axis" of sorts that denotes the columns of the line.

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
ATOM    145  N   VAL A  25      32.433  16.336  57.540  1.00 11.92      A1   N
```

The fields that are of importance for this assignment are:

Record name (Cols. 1–6) If this field is not equal to `'ATOM  '` than the line should be ignored.

Atom serial number (Cols. 7–11) The serial number of the atom (145 in the example).

Atom name (Cols. 13–16) The name of this atom (`N` in the example).

Residue sequence number (Cols. 23–26) The residue sequence number of this atom (25 in this example).

X coordinate (Cols. 31–38) The x-coordinate of this atom in angstroms (32.433 in the example).

Y coordinate (Cols. 39–46) The y-coordinate of this atom in angstroms (16.336 in the example).

Z coordinate (Cols. 47–54) The z-coordinate of this atom in angstroms (57.540 in the example).

Required Components [100 points]

1. Implement `readPDBFile` function. [20 points]
2. Implement `drawCA` function. [20 points]
3. Implement `hBonds` function. [20 points]
4. Read in `7hvp.pdb`. [5 points]
5. Draw Ca backbone for 7hvp. [5 points]
6. Display number of hydrogen bonds found in 7hvp. [5 points]
7. Read in file `1gfl.pdb`. [5 points]
8. Draw Ca backbone for 1gfl. [5 points]
9. Display number of hydrogen bonds found in 1gfl. [5 points]

10. Publish script and include introduction, code, results, and conclusions. [5 points]
11. Submit commented, clearly written code with function names that coincide with those supplied to you. [5 points]