Protein Modelling and Analysis with Python

Emma Gheysen
emma.gheysen@student.kuleuven.be

James O'Reilly
james.oreilly@student.kuleuven.be

1 Introduction

In this assignment we will parse a text file describing a protein structure and do some simple calculations on the atomic positions. Obtaining the three-dimensional structure of a protein provides important clues to how the protein functions and interacts with small molecules, DNA and other proteins. The text file comes from the Protein Data Bank, the standard database for protein structures. The PDB is an open repository for experimentally determined structures of proteins.

In this assignment we will parse the file and retrieve part of the information, particularly the resolution of the experiment and the atomic coordinates. Then we will explore the distribution of hydrophobic residues relative to the core of the protein, compared to the other residues. Finally, our program program will calculate which amino acid residues are closest to the ligand of the protein.

2 Parsing the File

The `.pdb` file used in this assignment contains the structure for the sperm whale myoglobin. The position of the protein's main ligand, the heme group, is also determined. Side note: Myoglobin was the first protein whose structure was determined. In 1958, Max Perutz and John Kendrew determined the 3D structure of myoglobin by X-ray crystallography. Four years later, they both received the Nobel Prize in chemistry for this innovation.

Before we discuss how this file is parsed, a quick description of the `.pdb` format is necessary. Effectively, `.pdb` files are text files which contain multiple columns (or fields) separated by whitespace. Each field occupies a specific position within each line of the text file. For example the "residue name" field occupies characters 18-20 in each line. There is a varying amount of whitespace surrounding each entry, as each entry need not occupy the entire character space given to that field. To account for this, the `strip()` function is used throughout our program, which removes any whitespace surrounding entries retrieved from the file.

Firstly, in the interest of keeping our program modular and our functions readable, we implemented a `retrieve` function which, given a file name and a list of fields to be retrieved, returns an array just with those fields. This allows us to easily access fields throughout the program. Note in Listing 1 that when parsing serial numbers, we converted the output to an `int`. Doing this for each numerical value retrieved from the file makes tasks like comparison and sorting

much more intuitive.

```
def retrieve(filename, fields):
    '''
    in: name of file and a list of fields as strings.
    out: list of fields from file
    '''
    output = []
    if 'atom' in fields:
        atomField = []
        with open(filename,'r') as infile:
            for line in infile:
                parsed_line = line[0:7].strip()
                atomField.append(parsed_line)
        output.append(atomField)

    if 'serial' in fields:
        serialField = []
        with open(filename,'r') as infile:
            for line in infile:
                parsed_line = int(line[7:11].strip())
                serialField.append(parsed_line)
        output.append(serialField)
                                        .
                                        .
                                        .
```

Listing 1: A snippet from the `retrieve` function. The logic shown above applies for retrieving each of the fields.

Listed below are the other functions used for parsing the original `.pdb` file. Combining these functions gives us a flexible workbench for parsing `.pdb` files.

- `aa_split()` is used to split a given file into two files based on the hydrophobicity of the entries, producing one file with hydrophobic entries, and one file with hydrophilic entries.
- `atom_split()` Takes in a full .pdb file, and produces a file with only atom entries, only hetatom entries, or both, based on a secondary argument given to the function.
- `get_coordinates()` Takes in a .pdb file, and returns an array of coordinate vectors for each of the entries. There is also an optional argument to only return the coordinates of hydrophobic or hydrophilic elements.

These functions can then be used to complete the first task in the assignment: to build a function named `parsePDB` which returns a list of the coordinates of the alpha carbon atoms, along with their residues, and also a list with the entries for all heterogens, the atoms that are associated with the protein during the crystallography, but that are not an integral part of it. This function is included in Listing 2.

```
def parsePDB():
    coordinates = get_coordinates(atom_file)
    residues = retrieve(atom_file, 'resName')[0]
    het_atoms = []

    with open(hetero_file,'r') as infile:
        for line in infile:
            het_atoms.append(line)

    output = list(zip(coordinates, residues)) + het_atoms
    return output
```

Listing 2: The `parsePDB` function.

3 Atom Geometry and Residue Hydrophobicity

In order to perform analysis on the structure of this protein, we must first analyse the positions of each of the alpha-carbon atoms. The positions are plotted in a three-dimensional scatter plot, to first get an overall feel for how these atoms are spread in three dimensional space.

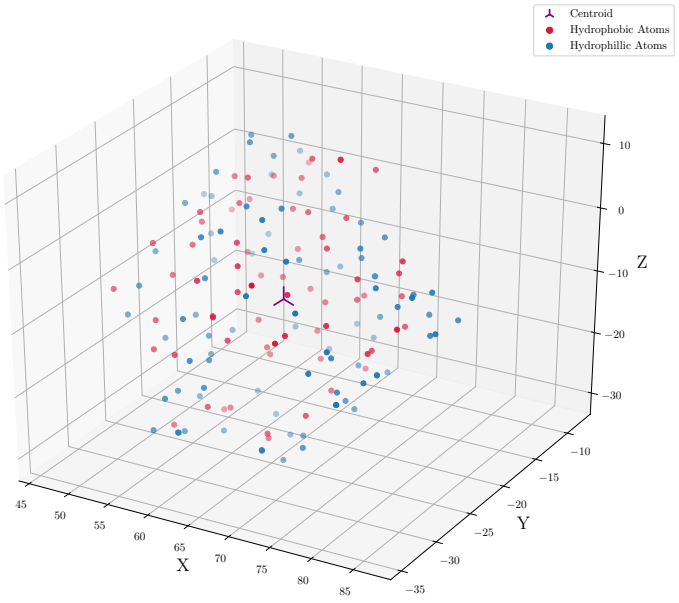


Figure 1: A three dimensional scatter plot showing the configuration of alpha carbon-atoms in the protein.

3.1 Computing the distances to the centroid

A function was built which, given a set of atom instances, finds the centroid and then calculates the distance to this centroid for each instance. For each instance this function outputs the residue name, the distance from the centroid, and whether it is hydrophobic or hydrophilic. It also outputs the centroid, which is useful further on in our analysis.

```
def distance_to_centroid(filename):

    distances = []
    markers = []
    residues = retrieve(filename, 'resNames')[0]

    coordinates = get_coordinates(filename)
    centroid = np.array(np.mean(coordinates, axis = 0))

    for coordinate in coordinates:
        distances.append(np.linalg.norm(centroid-coordinate))

    for residue in residues:
        if residue in hydrophobic:
            markers.append('PHOB')
        else:
            markers.append('PHIL')

    output = list(zip(distances, residues, markers))
    return output, centroid
```

Listing 3: A function for computing the distances for each atom from the centroid.

3.2 Analysing the distance data

The distances from the centroid for both the hydrophilic and hydrophobic atoms were then plotted on histograms and with a box plot (see Figures 2 and 3). It is clear from the box plot that overall the hydrophobic amino acids are closer to the centroid on average. This confirms the general hypothesis that hydrophilic residues tend to be more on the outside of the overall myoglobin structure where they interact with the polar solvent, which is the cytoplasm of muscle cells. These results also conform with the finding that the center of the protein consists of a hydrophobic cleft that complexes the heme group, suggesting we should find more hydrophobic residues closer to the center.

The separation between hydrophobic and hydrophilic amino acids is not distinct, however. the histograms in Figure 2 clearly overlap, and the closest amino acid residues are not exclusively hydrophobic. This again fits with our knowledge of how myoglobin is structured. Myoglobin is composed of eight alpha-helices connected by loops. In general, an alpha-helix has two 'faces': a more polar one facing the solvent and a less polar area that faces the center of the protein. If alpha-helix A and its residues are located closer to the center of the protein than alpha-helix B, then there are hydrophilic residues A which are closer to the center than hydrophobic residues of B. Also, there is a single hydrophilic residue occurring very close to the centroid. This can be explained by assuming it is a histidine residue as the Fe atom of the heme group is directly bound to a His residue of the amino acid chain.

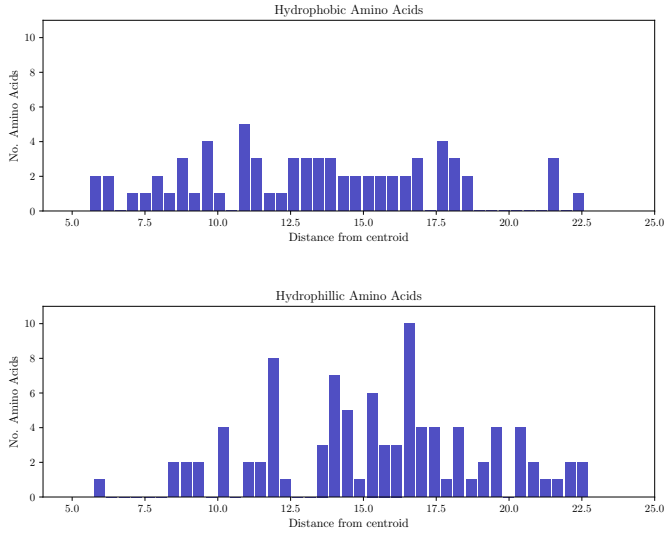


Figure 2: Histograms for distance of the hydrophobic and hydrophilic residues to the centroid.

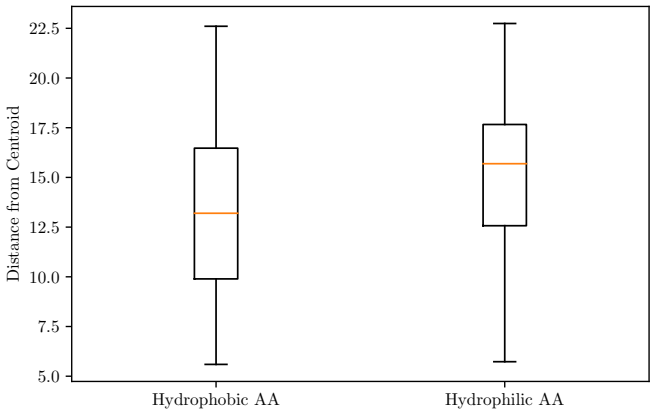
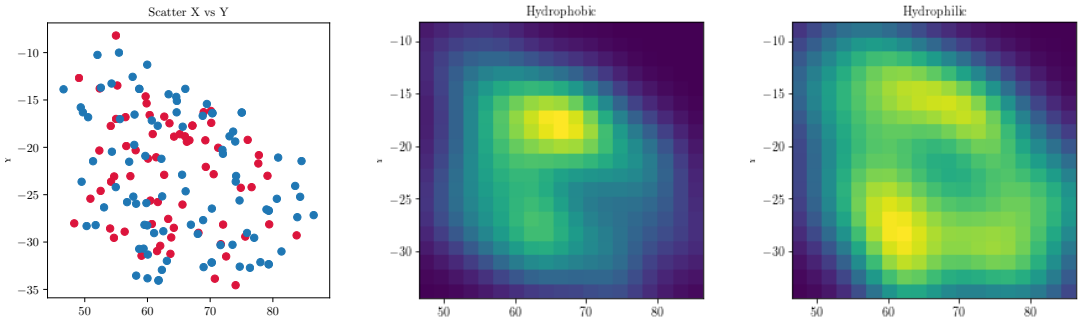


Figure 3: A boxplot showing the means and variances of the distance of hydrophobic and hydrophilic amino acids

3.3 Visualising distribution using Kernel Density Estimation

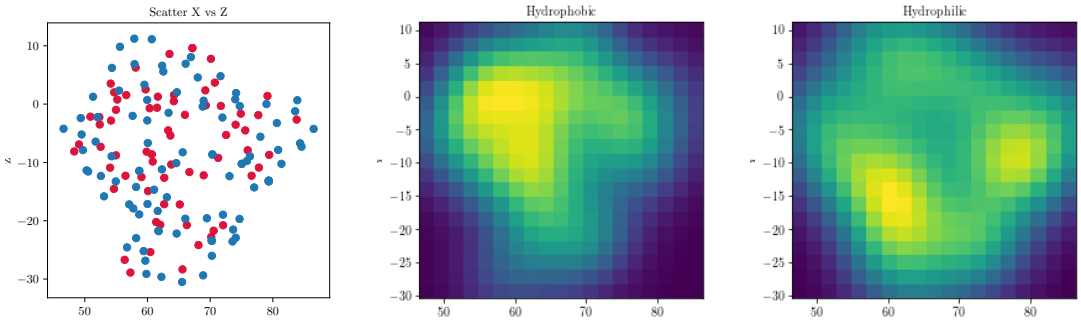
In order to effectively visualise the distribution of hydrophobic and hydrophilic atoms, the atom coordinates were flattened in each of the dimensions and plotted on two dimensional scatter plots shown below. The hydrophilic atoms tend to gather at the edges of this cluster, and the hydrophobic atoms are clustered toward the center. A two dimensional kernel density estimation (KDE) was then calculated for each of the two dimensional data sets and then plotted with a colour gradient, to better illustrate how the atoms were clustered. These KDE plots show that

the hydrophilic atoms are clustered around the outside of the protein, and show the position of the hydrophobic cleft into which the heme group is inserted.



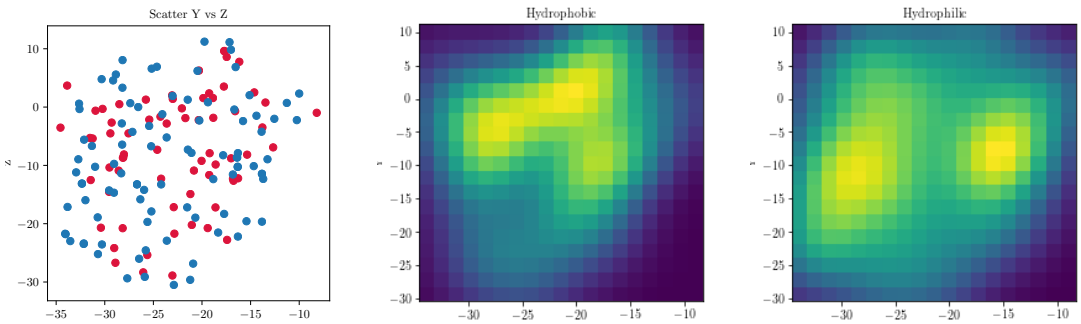
(a) (b) (c)

Figure 4: Scatter plot and kernel density estimate plots for the x and y coordinates.



(a) (b) (c)

Figure 5: Scatter plot and kernel density estimate plots for the x and z coordinates.



(a) (b) (c)

Figure 6: Scatter plot and kernel density estimate plots for the y and z coordinates.

4 Amino acid environment of the Fe atom

We created a function that takes the list of heteroatoms and the name of an atom, returning a list of position coordinates (see Listing 4. Using this function, the position coordinates of the Fe atom can be retrieved. This iron atom is part of the heme ligand which is located in the central cleft of the myoglobin protein.

```
def search_heterogen(filename, hetatom_name):
    coordinates = []
    with open(filename,'r') as infile:
        for line in infile:
            if line[12:16].strip() == hetatom_name:
                coordinates = [float(line[30:38]), float(line[38:46]), float(line
    [46:54])]

    return coordinates
```

Listing 4: The `search_heterogen` function.

4.1 Nearest neighbours

A function was written that returns a specified number of amino acids which are closest to the Fe atom, represented by their alpha-carbon coordinates. The function returns the coordinates of the alpha-carbon, the unique serial index, the distance to the centroid, and the residue type.

The function takes in a filename, the coordinates of the atom to which the distances will be calculated, and the number of closest neighbours we want to find. Each atom entry in the file is first sorted in a list by its distance to the atom of interest. This list is then truncated based on the required number of neighbours. However, if the list contains the input atom itself, we do not want to include this atom in the list of neighbours. so first we run a check to see if it is in the list,and then if one wants the closest n neighbours, then we must truncate the list at $n + 1$. Therefore, to find the five amino acids which are closest to the Fe atom, we give the function the file containing atoms, the coordinates of the Fe atom, and the number five as input arguments (see Listing 5).

```
fe_coords = search_heterogen(hetero_file, 'FE')

atom_neighbours("atomfile.txt", fe_coords, 5)
```

Listing 5: Using the `atom_neighbours` function to find the five alpha-carbon atoms which are closest to the Fe atom.

4.2 Neighbours of the Fe atom in myoglobin

The five closest amino acids to the Fe atom are VAL, HIS, HIS, LEU and ILE (See Table 1). These results were verified with existing results in the Protein Data Bank, by visualising the three dimensional structure of the atom and identifying the residues nearest to the heme group [1]. The iron atom is bound to four nitrogen atoms that are part of the heme group, to one oxygen atom which composes part of the carbon monoxide molecule in this structure, and to a histidine residue of the myoglobin protein. As there are two histidine residues, one can look at

the actual structure to figure out which residue is bound to the heme group, and also to confirm it is a histidine residue. The amino acid that is bound to the Fe atom indeed is a histidine, with index 94. The other histidine residue with index 98 is part of a nearby loop connecting two alpha helices. The remaining three amino acids are hydrophobic which fits with our understanding that the heme group is integrated in a hydrophobic cleft in the centre of the protein.

Table 1: Fe atom neighbours.

Coordinates	Serial No.	Residue	Residue No.
(63.792, -29.514, -10.362)	571	VAL	69
(75.875, -29.043, -9.742)	757	HIS	94
(77.02, -29.553, -14.29)	794	HIS	98
(72.547, -31.521, -5.296)	729	LEU	90
(76.607, -24.205, -12.283)	819	ILE	100

4.3 Visualising nearest neighbours

As an extension, we built a function which allows us to view a given atom and how it is positioned in relation to its neighbours in three dimensions. It takes in a file name, the serial number of the atom of interest, and the number of neighbours to be plotted. An example is included in Listing 6 and Figure 7. This function could be included to colour each scatter point with a colour specific to each amino acid residue, as well as showing the distances, which could be useful for visualising clusters of amino acids within proteins and finding familiar patterns related to secondary structure.

```
1 plot_neighbours(atom_file, 29 , 5)
```

Listing 6: Using the `plot_neighbours` function to plot the five alpha-carbon atoms which are closest to a given atom with serial number 29.

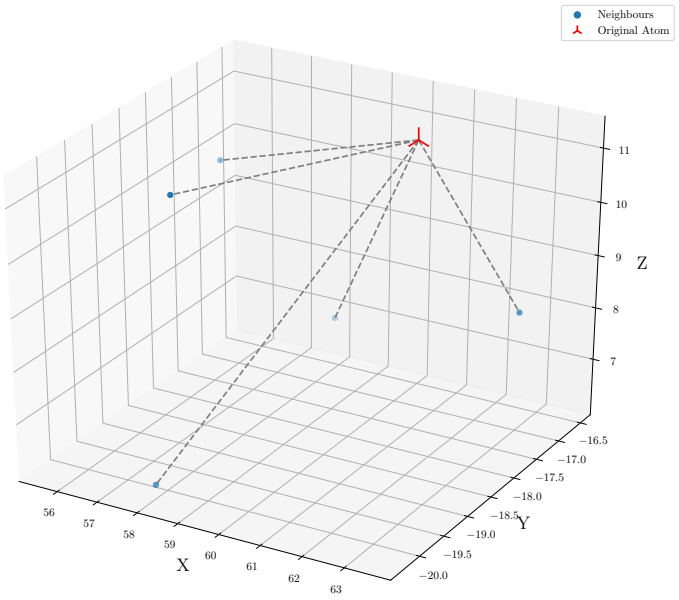


Figure 7: A three dimensional scatter plot showing the configuration of atom neighbours.

References

[1] Rscb protein database: 5kkk sperm whale myoglobin. `https://www.rcsb.org/structure/5kkk`. Accessed: 2019-12-7.