Project 2: Water Chemical Potential

1 Introduction

The function of this python module is to compute the volumetric map of the chemical potential for water given a set of molecular dynamics trajectories. To achieve this, first it calculates the frequency of water molecules over a cubic grid of size 1 Å for each dimension, in a 3D array H. This values are normalized to estimate probabilities from them and with this probabilities the chemical potential is estimated using the formula $G(x, y, z) = -k_B T \log(H(x, y, z))$, where $k_B = 0.001987191 \frac{kcal/mol}{K}$ and $T = 298K$. Finally from this array G a cube file is generated.

2 Usage

The module must be called as:

```
python waterpot.py data-folder
```

The parameter data-folder is the path, absolute or relative, where all the data is stored. The code assumes that the data is stored in sub-folders.

3 Implementation

The code makes intensive use of the numpy and HTMD libraries, although others are used at some point, such as: sys, copy, glob and scipy.ndimage. The core of the code is the Simulation class. This class loads the data from the trajectories, calls the methods align and wrap of HTMD.Molecule and has two additional methods, prepare_coords, which extracts the coordinates of the water atoms for all frames in a numpy 3D array and it reshapes the array to a n x 3 array, where n is the product between the number of water atoms and the number of frames. The other method is the water_hist, which calls the prepare_coords method and then calculate the the histogram of the coordinate matrix over the bins that are previously calculated.

To calculate the bins we need to analyse one of the structures we have, and since the structures must be very similar, we only need to calculate it once and we can reuse it. In order to do this, a new class SimulationRef is created. SimulationRef inherits from Simulation, its only difference is the constructor. When creating an instance of the SimulationRef class, in addition to loading the structure and aligning it, we calculate its maximum and minimum position, which after adding a certain distance will serve as the

start and end of the histogram bins. We just create an array from the starting point to the ending point with a step of 1, and store it to pass it to the histogram. Once this is done the water_hist method is called and we have the occupancy vector for the first structure. The SimulationRef object created is passed as an argument when creating a Simulation object, in order to copy the information needed to compute the histogram.

The code will detect how many subfolders there are in the folder specified, use the first one as reference and then iterate over the rest, creating a Simulation object for each subfolder and adding the result of the histogram to the previous value. Once the iteration finishes, the array with the accumuled histogram counts is normalized and averaged over all the structures, so that it corresponds to a probability and very small quantity is added to avoid problems with the logarithm. Once we have calculated G, we apply a gaussian filter from the scipy.ndimage module. This filter allows us to distribute the occupancy of a water oxygen over neighbouring grid. Finally, we call the htmd function writeVoxels to obtain a cube file to visualize the chemical potential and write the reference structure to a pdb file, called ref_structure.pdb, in order to be able to visualize appropriately the isosurface. This pdb file is necessary because the molecules are moved by the code, and therefore the original pdb will not allows us to visualize the isosurface properly, since they both will be moved.

4 Results

As mentioned before, the code makes an intensive use of numpy. In fact, almost every computing instensive part of the code is offloaded to numpy. The reason of this is obvious, the lower level operations performed in numpy make the code much faster than explicit computations in Python.

In order to evaluate the results, we can view the reference molecule in VMD, and load the cube file into the molecule. In figure 1 the result of this process can be seen. This figure is obtained by superposing three representations, one with points for the water molecules, one with NewCartoon for the protein, and a isosurface for the cube file. The isosurface is represented with isovalue of 7.5, step 2 and drawing method wireframe. As can be seen, there are two regions in the isosurface, the internal one is the one of interest. We can see that it fits quite well the protein, although in the regions around both N and C terminus the protein goes out of the isosurface.

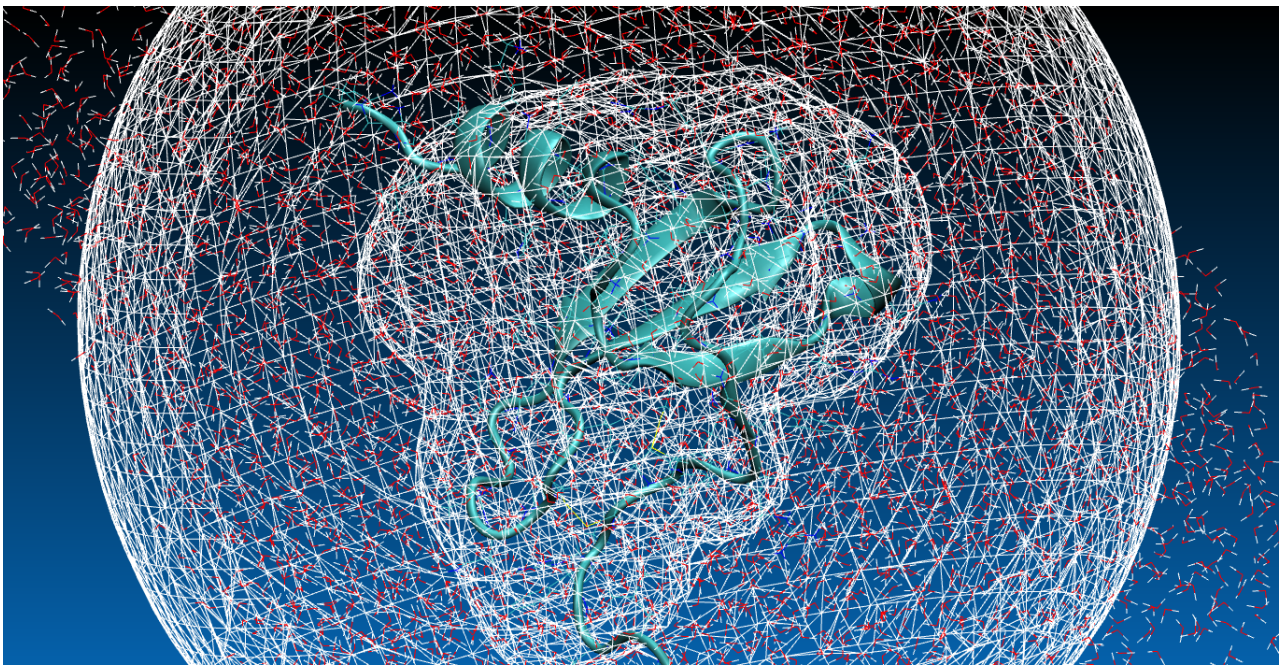

Figure 1: Snapshot of the protein embedded in water molecules and the isosurface resultant