Jeanna Goodrich Balreira

Vanessa Moreno

May 10, 2013

Numerical Analysis 3351-1

Dr. Nguyen

FINAL REPORT ON VORONOI TESSELLATIONS, VORO++ AND VMD

INTRODUCTION: GOALS OF THE PROJECT

Our group comprised of three members, Valeri Alexiev, Jeanna Balreira, and Vanessa Moreno, worked on the semester-long project, *Analyzing the structures and dynamics of protein/lipid interactions in Lipid Nanodomains using a deterministic centroid Voronoi tessellation method,* under the guidance of project mentors Dr. Hoa Nguyen and Dr. Kelvin Cheng. Our goal was to develop a Voronoi tessellation [VT] method to calculate the distributions and kinetics of the surface area, volume, and void spaces of lipid and protein groups. The motivation behind our goal is to help develop a method to better calculate surface area, volume, and voids of biological molecules that are important to in characterizing the protein misfolding and aggregation on cell membranes (Cheng).

Our group utilized the imaging and analysis software Voro++, VMD, and Voroprot. Valeri focused on understanding, using, and extending Voroprot. This report will describe Jeanna and Vanessa's focus on the data mining and statistical analysis of the output of Voro++ and applying the output into VMD for visualizations.

METHODS: VT THEORY AND 2D PICTURES TO ILLUSTRATE TYPES OF VT'S

The first stage of our project was dedicated to research. The first struggle we encountered was lack of strong biology and chemistry backgrounds. In order to accomplish our goal, we need to first understand it. Using the Internet and related published articles and papers we sought information about the structures (proteins, misfolding, lipids, etc.) we were modeling and analyzing. We learned proteins are biological molecules made from chains of amino acids whose vast amount of functions include the replication of DNA, catalyzation of metabolic reactions, and transportation of molecules from one location to another. The folding of a protein is a specific three-dimensional structure that determines the activity of the protein. Thus, the misfolding of a protein can lead to toxic functionality of the protein.

Alzheimers disease [AD] is considered a protein misfolding disease (Poirier and Ross). AD is linked to the accumulation of beta-amyloid deposits, a misfolding of a protein that builds plaque on the cell membrane. AD is the disease the team of faculty leading this research focuses on. As stated earlier, the calculations of surface area, volume, and voids of molecules help characterize protein misfolding and aggregation on the cell membrane. The methods traditionally used to calculate these parameters are based on assumptions of the volume of molecules, estimations of the surface area, and a disregard for voids. Annular lipids surround and embed the protein, this makes for many errors in the calculation of the parameters. We believe a VT method will more accurately calculate the distributions and kinetics of these parameters in a protein (Cheng).

Our next step was to understand VT theory as well as the types of VTs we would be working with. After searching through the code, we found that Voroprot creates tessellations using the additively weighted VT method and Voro ++ uses the power VT method, sometimes referred to as radical VT method. We will now provide definitions and examples of the VTs used.

Definition: Voronoi Tessellation / Voronoi Diagram
Given a plane with N points, and a set of generating points, we partition the plane into convex polygons such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generation point that to any other (Wolframalpha).

Definition: Euclidian VT
Given a plane with N points, and a set of generating points, we partition the plane into convex polygons such that each polygon contains exactly one generating point and every point in a given

polygon is closer to its generating point than to any other. Here, "closer" is determined by the smallest Euclidian distance from one point to any of the generating points (Wolframalpha).

Definition: Power VT (Also referred to as Radical VT)

$$D\ (p_i,\ w_i;\ p) = [\ ||\ p-p_i\ ||\ ^2\ ] - w_i^2$$

Given a set of circles, with the center located at p_i and a radius of w_i, we create a partition of the plane into cells by assigning a point, p, to the region of $p_i;w_i$ for which the power distance to $p_i;w_i$ is smaller than the power distance of any other generator. Here, $p_i;w_i$ are generators and p is any point on the plane.

Definition: Additively Weighted VT

$$D\ (p_i,\ w_i;\ p) = [\ ||\ p-p_i\ ||\ ] - w_i$$

Given a set of circles, with the center located at p_i and a radius of w_i, we create a partition of the plane into cells by assigning a point, p, to the region of $p_i;w_i$ for which the additively weighted distance to $p_i;w_i$ is smaller than the additively weighted distance of any other circle. Here, $p_i;w_i$ are generators and p is any point on the plane.

To further understand the VT methods used in Voroprot and in Voro++, we created a small-scale Voronoi tessellation algorithm in a two-dimensional plane with two generators. We let the generators be points $p_1=(0.25,\ 0.25)$ with radius $w_1=0.25$ and $p_2=(0.75,\ 0.75)$ with radius $w_2=0.5$. With our domain on the Cartesian coordinate system and both our x- and y-axes ranging from 0 to 1.5 with increments of .01 units, we used meshgrid in MATLAB to create our plot. Our input is the x and y coordinate position of the center of p_1 and p_2 and the radius, or weight, of each point. Our points p_1 and p_2 then respectively become circles where $C_1 = 2\pi w_1$ and $C_2 = 2\pi w_2$. Using a loop, our script runs through all the points in the meshgrid and calculates the meshgrid's point's power distance from C_1 and C_2. If a point's power distance to C_1 is less than the power distance to C_2, then that point is assigned to the region of p_1, and vice versa. If there is a case where the power distance to C_1 equals the power distance to C_2, then the point lies in the boundary. The same procedure is done for the additively weighted VT.

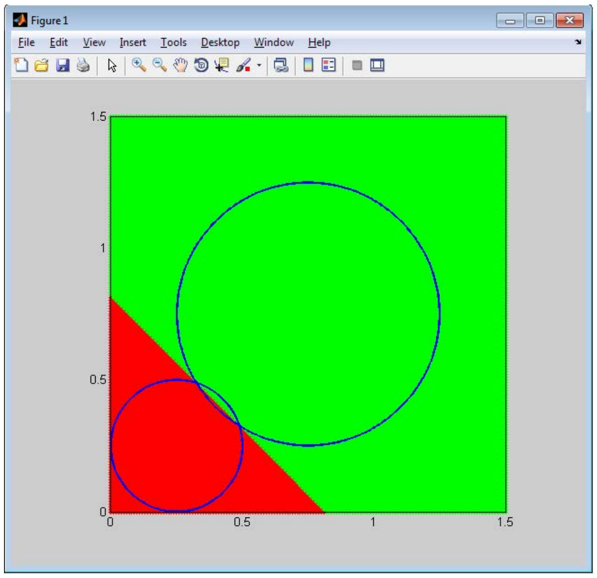


Figure 1: Power VT with 2 generators

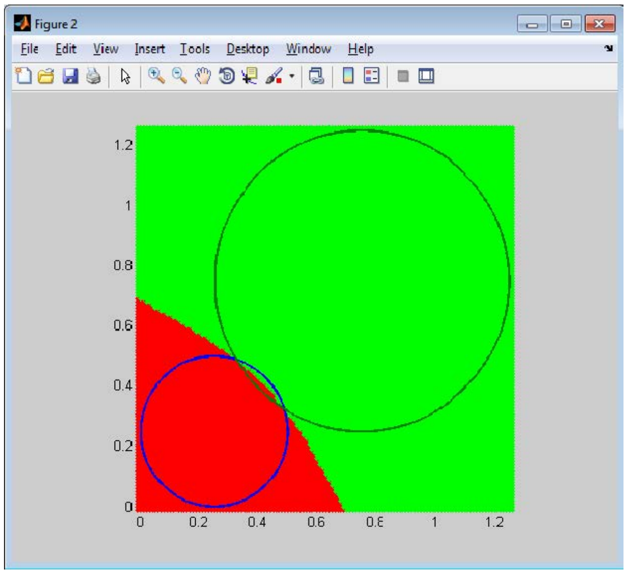


Figure 2: Additively weighted VT with 2 generators

We note that the partition of the plane in the power VT is linear, while the partition of the plane in the additively weighted VT is curved. The power VT distance from point p to the generator pi is calculated using $D\ (p_i, w_i; p) = [\ || p-p_i ||\ ^2\ ] - w_i^2$. The formula then simplifies to the linear function $D\ (p_i, w_i; p) = [\ (xp-xp_i)\ ^2 + (yp-yp_i)\ ^2\ ] - w_i^2$; this linear function is what causes the boundary line in the plane to be linear. The additively weighted VT boundary is curved due to the presence of the square root in the distance formula $D\ (p_i, w_i; p) = [\ || p-p_i ||\ ] - w_i$. We can also see differences in the overlapping region. Similarly, each circle's Voronoi region can account for some area in the overlap; however, the additively weighted VT concludes that a majority of the overlap region to belongs to the smaller radius circle, whereas in the in the power VT the circle with a greater radius accounts for a majority of the overlap.

To further understand the behavior of these two types of VTs with a greater number of generators, we extended our MATLAB script to create power and additively weighted VTs for "n" given generators. We created examples using generators p_n, where $n=5$: $p_1 = (0.25, 0.15)$, $p_2 = (0.45, 0.35)$, $p_3 = (0.5, 0.75)$, $p_4 = (1.25, 1.5)$, $p_5 = (2.25, 2.75)$ with corresponding weights $w_1=0.25$, $w_2=0.5$, $w_3=0.3$, $w_4=0.5$, $w_5=0.25$.

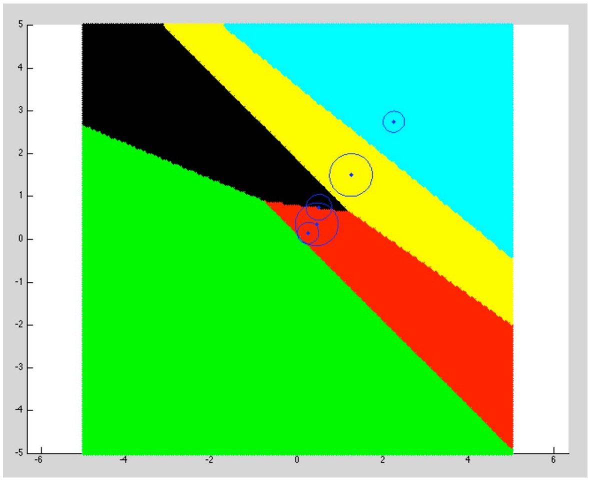


Figure 3: Power VT with n=5 generators

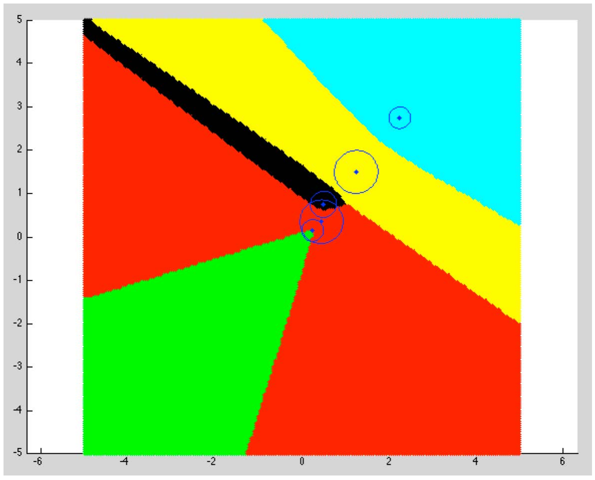


Figure 4: Additively weighted VT with n-5 generators

At this point, we knew how to create the types of VTs implemented in Voro++ and Voroprot; our next step was to make sure that our MATLAB-produced VTs were accurate. To check the accuracy of our code we needed to compare our results to the results of Matlab's Voronoi command on the same points. MATLAB's Voronoi command creates an exact Euclidean Voronoi tessellation, meaning the plane is partitioned using the distance formula $D\ (p_i, w_i;\ p) = ||\ p-p_i\ ||$ where p is any point and p_i is a generator. We modified our code by changing the power and additively weighted distance formulas to the Euclidian distance formula $||\ p-p_i\ ||$ and assigning the same weight, $w_i=0.25$, to all the generators.

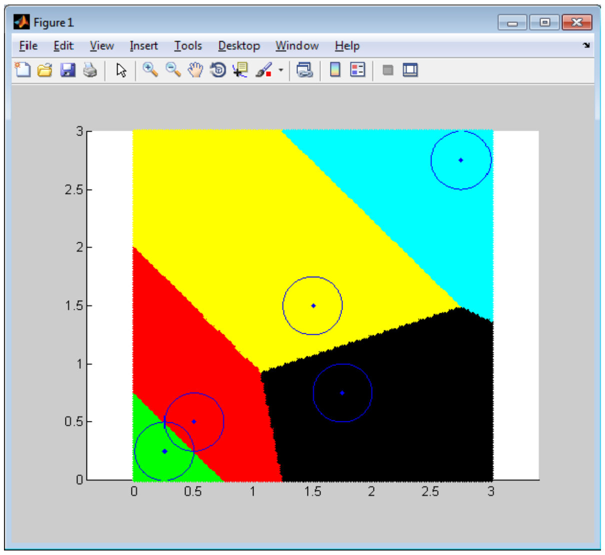


Figure 5: Euclidian VT with weight = 0.25

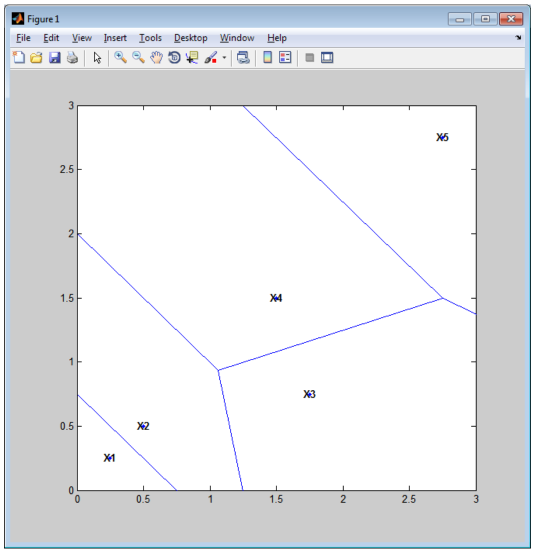


Figure 6: MATLAB's Voronoi command on same points

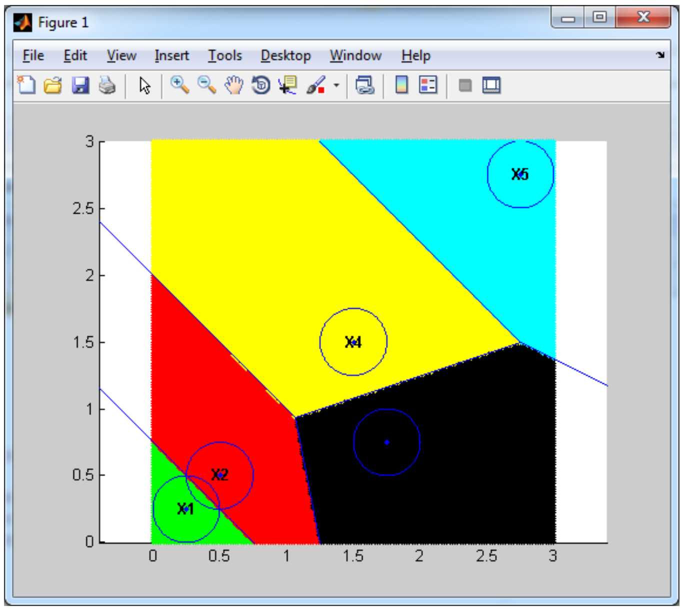


Figure 7: Figures 5 and 6 overlaid in the same plot

As shown by Figure 7, the partitions of the plane are the same shape for each Voronoi region. The resolution of our code improves to more accurately match MATLAB's rendition as we decrease the step size of our graph. Our understanding and calculations for power VTs and additively weighted VTs derived from the construction and application of our code now can be used to analyze greater structures in Voro++ and Voroprot.

RESULTS: VISUALIZATION IN VORO++ AND VMD

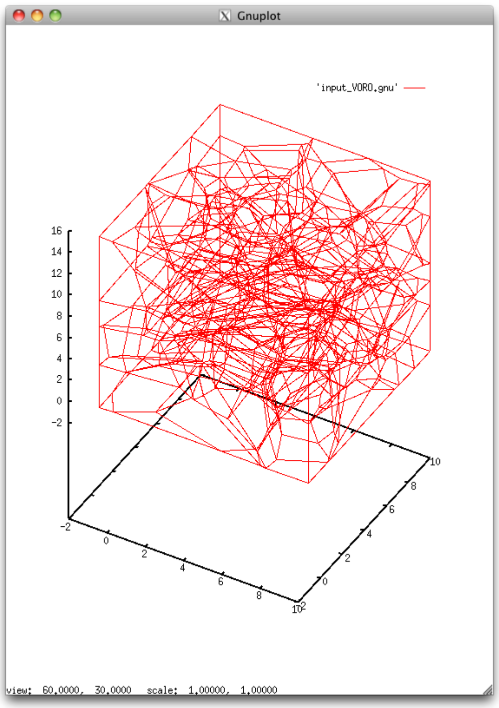
Once we were able to visualize a smaller sample of power VT and additively weighted VT, we could better apply our knowledge to large-scale visualizations in Voro++ and VMD. Voro++, a C++ library that carries out three-dimensional VT computations, can be used to both calculate and visualize three-dimensional containers of both non-weighted (Euclidian) cells and power or radical cells. VMD, a more high-powered visualization program, can take a fully compiled PDB file and display the protein structure as layers, which prove vital to visualizing nearest neighbor calculations—calculations that can differ based on the type of tessellation method that we employ.

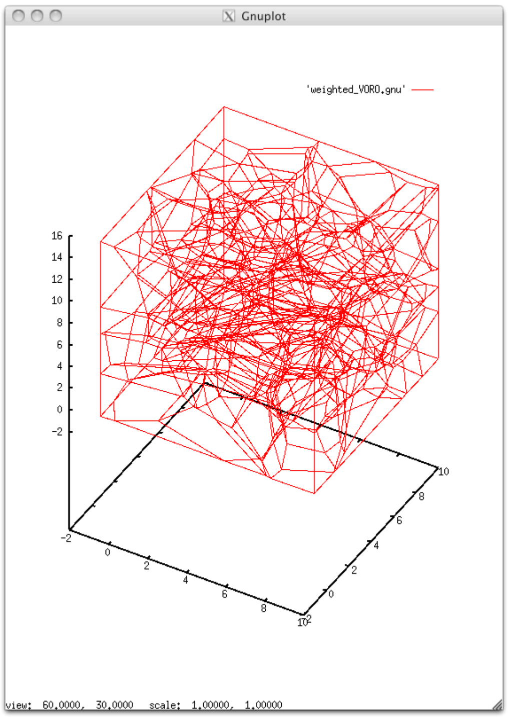
After completing our code for both 2 and *n* generators, we were able to see our power Voronoi tessellation methods come into play using Voro++. It was exciting to realize how our two-dimensional tessellation algorithms could potentially be applied in a three-dimensional setting. Of course the Voro++ library does not use our exact algorithm for mapping a power tessellation, but we were able to see the linear tessellations for each atom, rather than any sort of spherical shape that might appear using an additively weighted tessellation computation.

Voro++'s `radical.cc` compares two three-dimensional cubes packed with atoms. One cube does not take the atom's radius as input and therefore computes a normal Voronoi tessellation based on Euclidian distance; the other cube does utilize the atom's radius and computes a radical Voronoi tessellation based on the power distance formula (Rycroft).

While we already had a file to create the normal container, `input_VORO`, which provided us with the atom ID and its x, y and z coordinates, we needed a file that also had each atom's radius appended. Creating this file was a two-step process. First, given a file that matched an atom type with its radius, we needed to amend this file because the atom type ascii did not directly match the ascii from the PDB. After editing this file, Valeri Alexiv created a Python script that would strip the atom ID (in this case, the line number of the atom) its x, y and z coordinates from the PDB and then attach the matching radius from the radius file, based on the atom's type. The resulting file, `weighted_VORO`, could be used for generating the `radical.cc` radical container.

Compiling and executing `radical.cc` with these two input files generated the two desired containers, rendered below with GNUplot. A side-by-side comparison of a small subsection of these two containers showed little difference in the two plots:





Figures 8 and 9: GNUplot of C2_29 x, y, and z coordinates, with weights appended to Figure 6; subsection: x: 1 to 10; y: 1 to 10; z: 1 to 15; number of blocks: x = 5; y = 5; z = 5.

However, an overlay of the two subsections showed slight differences in the tessellations:

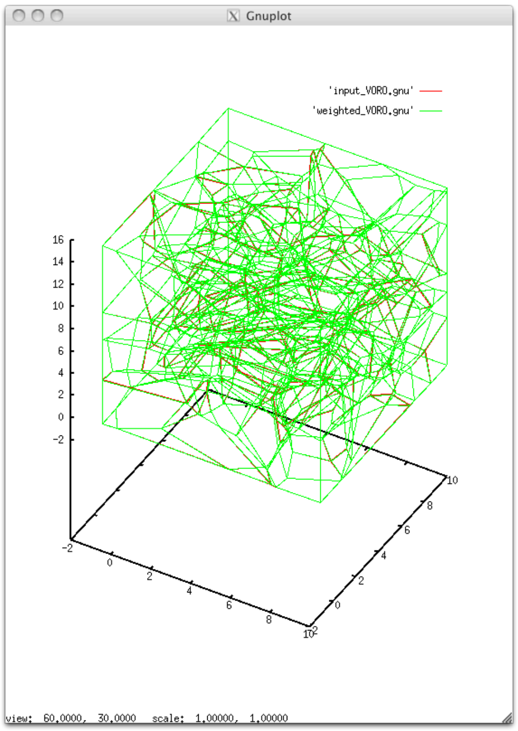


Figure 10: GNUplot overlay of Figures 8 (red) and 9 (green)
subsection: x: 1 to 10; y: 1 to 10; z: 1 to 15
number of blocks: x = 5; y = 5; z = 5

Differences occur where red is visible through green, showing the slight differences between normal and radical tessellation computations. As we travel out from the subsection, we can assume that there will still be differences in these tessellations. This visualization helps draw the conclusion that our results will most likely vary depending on the tessellation method we use to divide a container. We will discuss this further in our "Conclusions" section.

Using the Voro++ calculations on the C2_29 PDB, we generated a weighted PDB file and imported it into VMD. From the provided data in C2_29_weighted_VT, we first uploaded `protein.pdb`, the weighted C2_29 protein, into VMD. Next, we uploaded `POPCHOLSOL_1.pdb`, the first lipid shell of the protein. We selected a color, blue, to identify this lipid shell. We continued to upload the remaining eight lipid shells and color-labeled each one to distinguish each shell. The image below shows the final image produced by VMD once the protein and all nine lipid shells were uploaded.

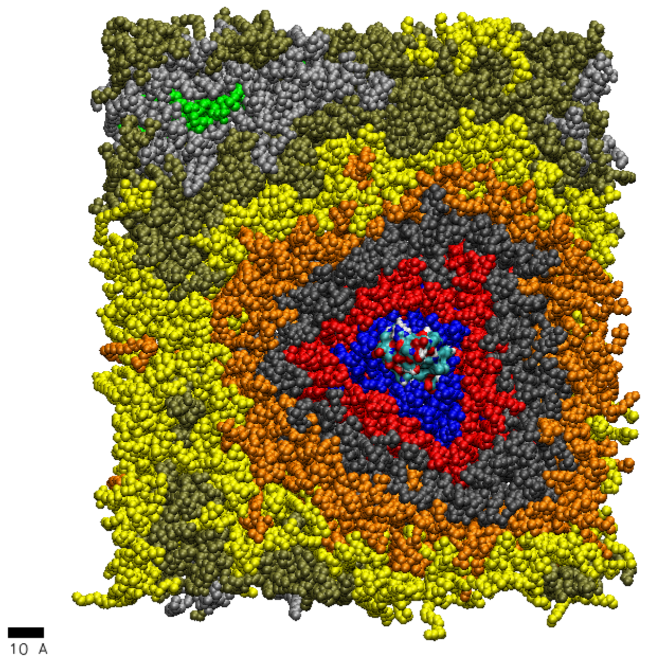


Figure 11: VMD rendering of protein and surrounding nine lipid shells

We were also given the interfacial water molecules for the C2_29 protein. We similarly modeled the interfacial water molecules in VMD. We first uploaded `protein.pdb`; however, this time we uploaded `SOL_1neigh.pdb`, the first solvent shell adjacent to the protein. Next, we uploaded the second solvent shell, `SOL_2neigh.pdb`, the solvent neighboring the first lipid shell. The third solvent shell is the solvent neighboring the second lipid shell, and so on. We uploaded all nine solvent shells into VMD and assigned an identifying color to each solvent shell that matched the first image. For example, the first solvent shell is adjacent to the protein; therefore, we assigned it a unique color. However, the second solvent shell is the solvent neighboring the first lipid shell, therefore we assigned it the same color, blue, as the first lipid shell. The image below shows the final image produced by VMD once all nine solvent shells were uploaded.

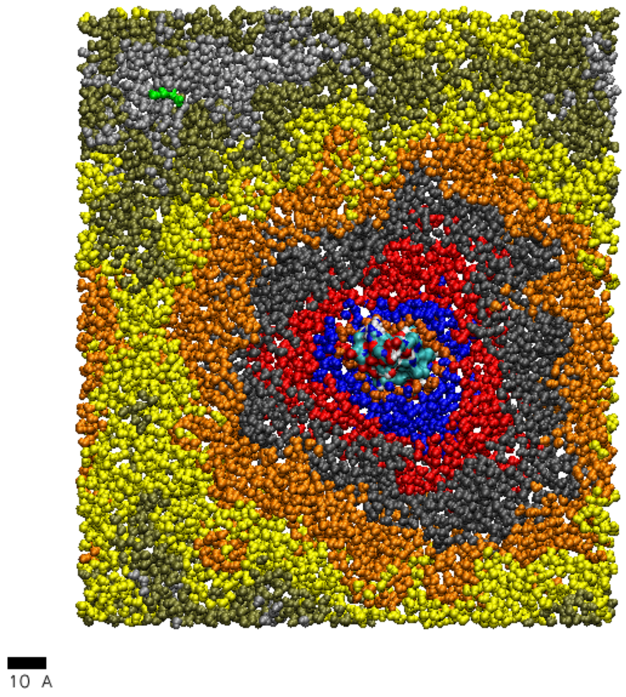
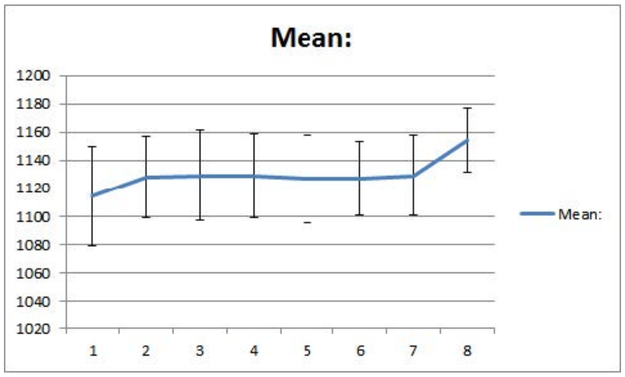


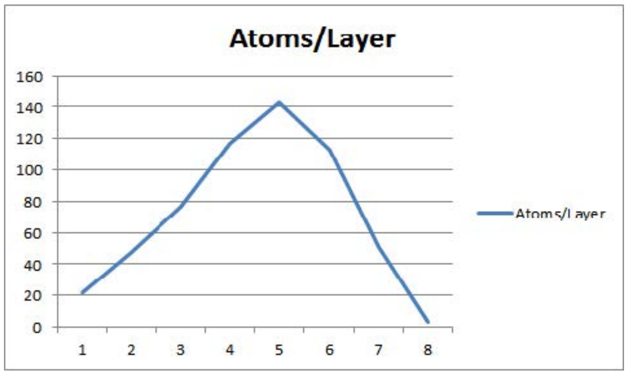
Figure 12: VMD rendering of protein and surrounding nine solvent shells

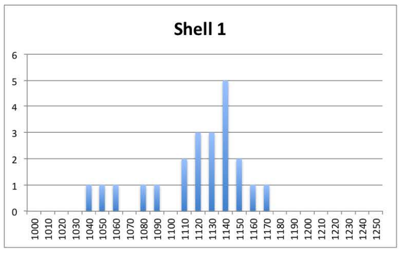
Based on these layer computations, we were able to generate a numerical comparison in addition to a visual one, calculating the number of molecules of different types present in each layer, plus their average volume within that layer. We then used histograms to display the number of molecules in a layer that fell in a range of volumes. We analyzed four different types of molecues: POP, CHO, CHO_O6, and SOL. For the histograms, the number of atoms present in a layer is represented in the bar, while the groups of volumes are represented as the bins.

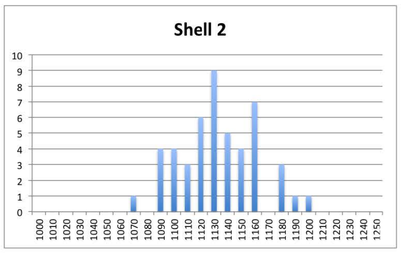
It is interesting to note that while the number of POP, CHO and CHO_O6 molecules decrease in the outer layers (past layer 5), the number of SOL molecules continues to increase. This suggests that more solvents are present in the outer layers, while more of the other molecules are concentrated more closely around the in the inner layers.

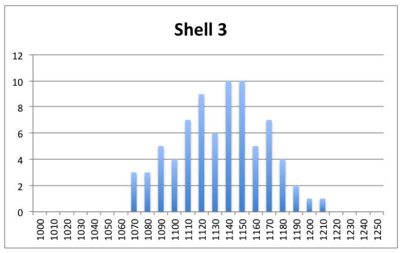
POP molecules

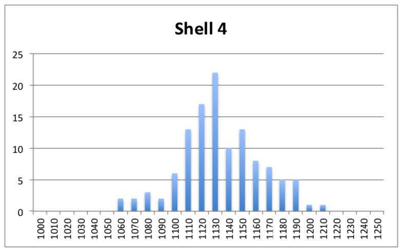


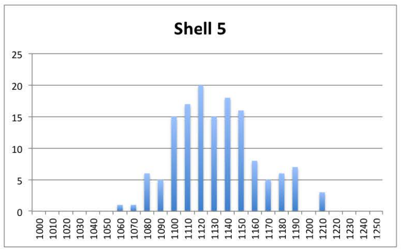


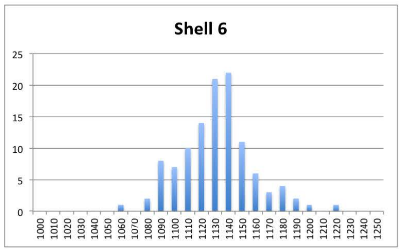


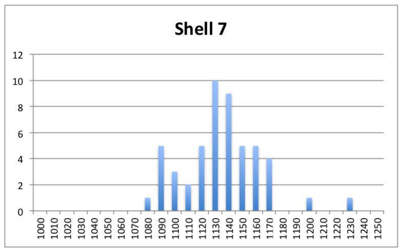


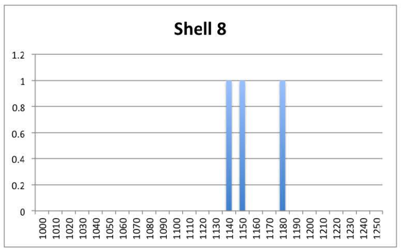


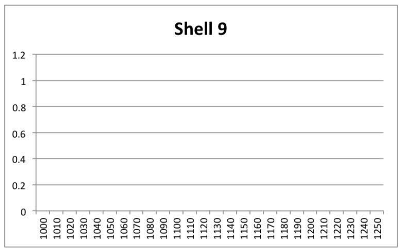




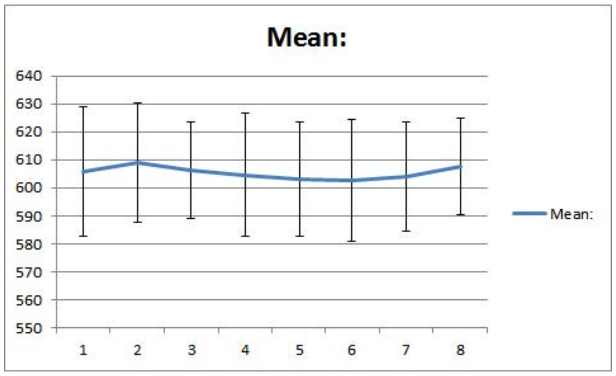


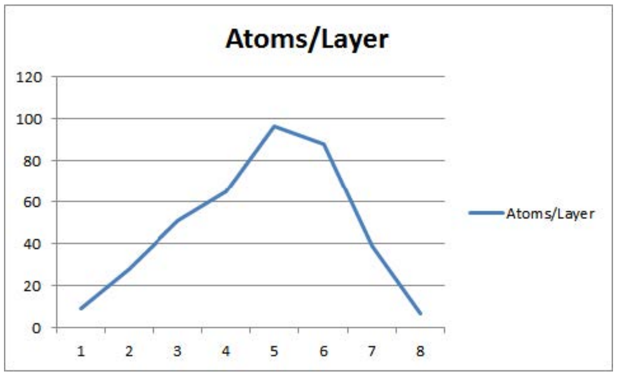


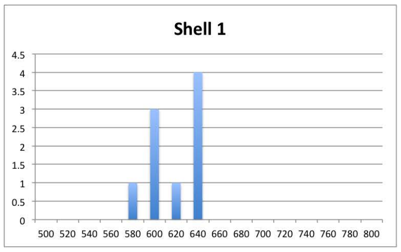


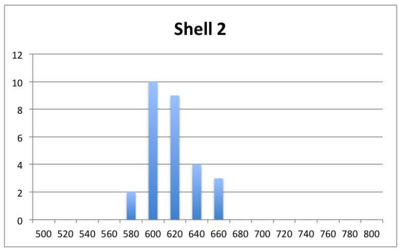


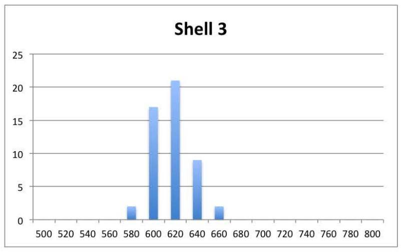
CHO molecules

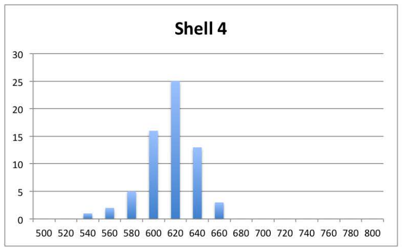


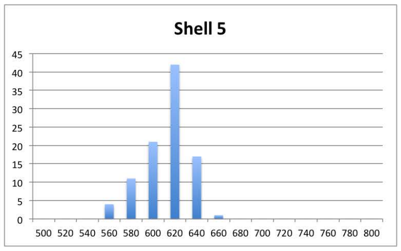


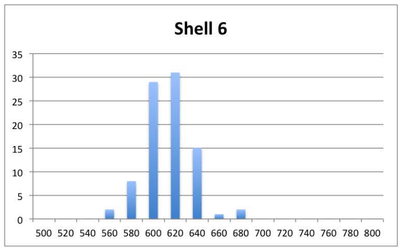


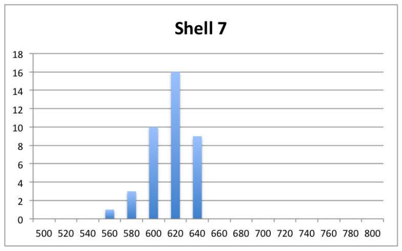


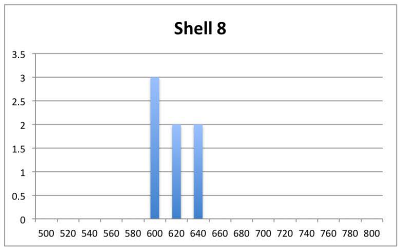


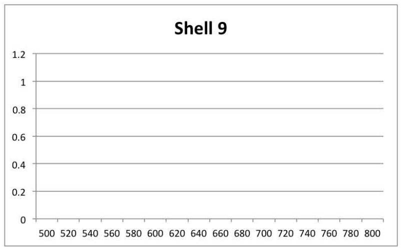




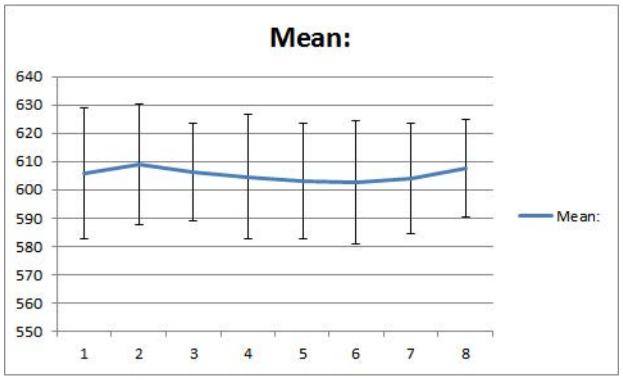


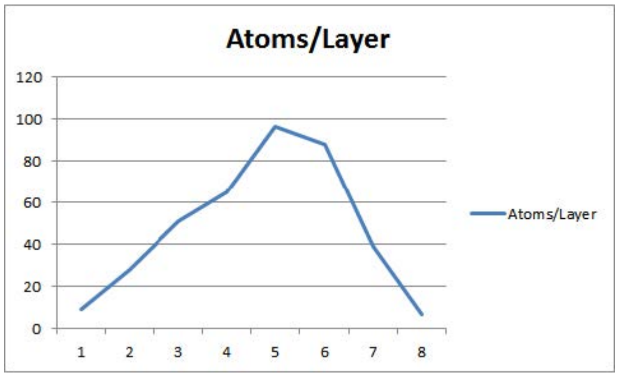


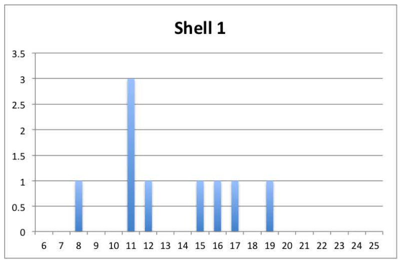


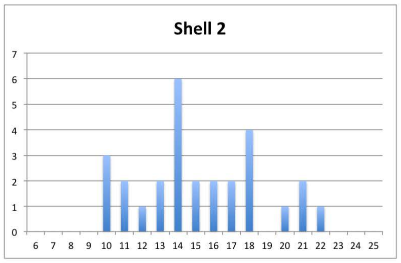


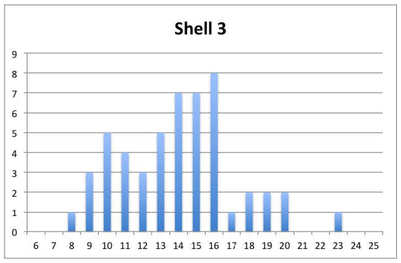
CHO_O6 molecules

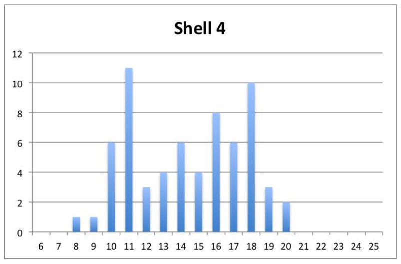


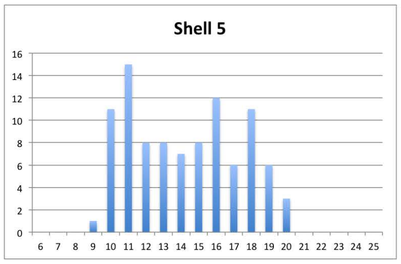


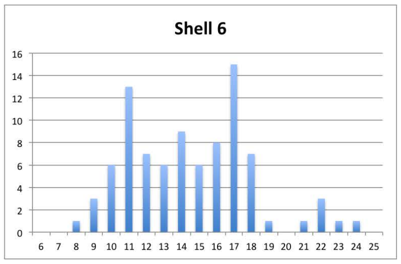


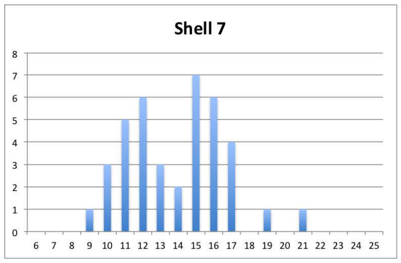


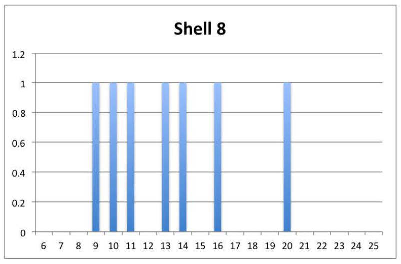


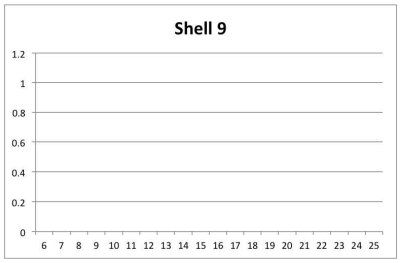




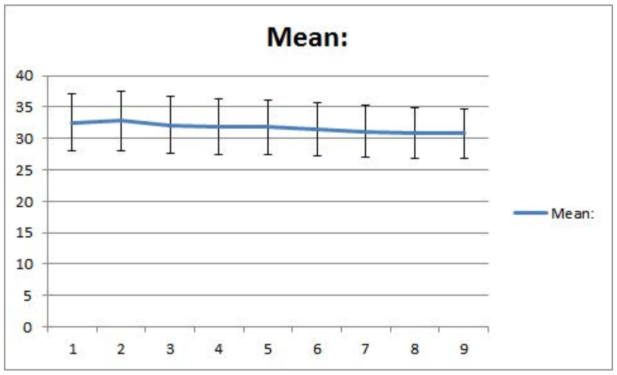


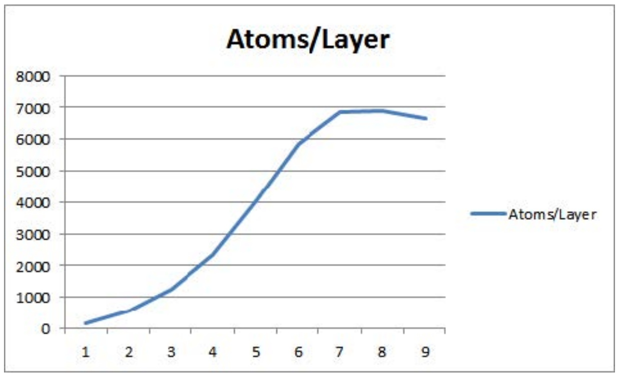


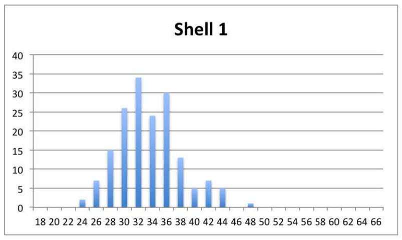


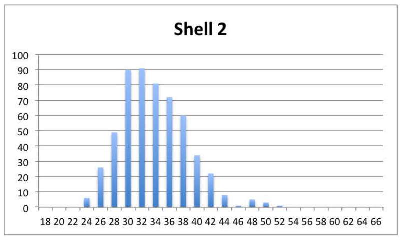


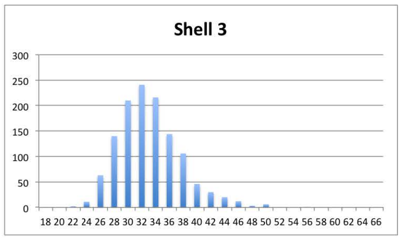
SOL molecules

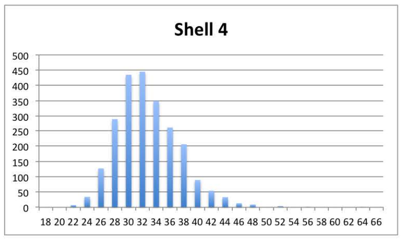


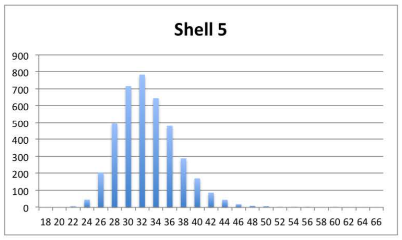


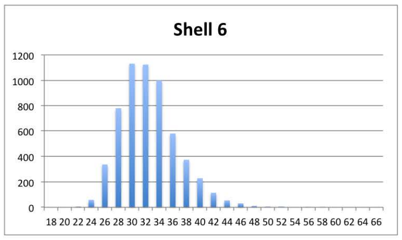


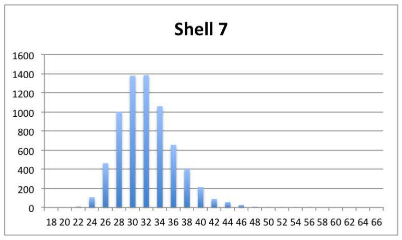


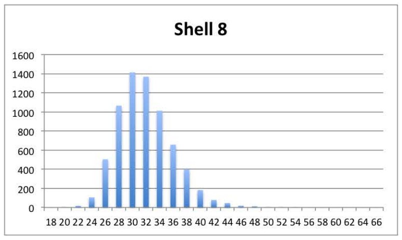


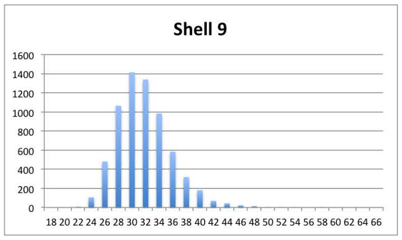












CONCLUSIONS AND FURTHER RESEARCH

Our Voro++ results showed us a quite interesting conclusion: the type of VT method we use (non-weighted, additively weighted, power) will have an impact on the composition of nearest-neighbor layers. This conclusion is computationally significant due to the fact that these tessellations are what help us identify the nearest-neighbor shells in proximity to the central protein. If the tessellations differ greatly enough, there extends the possibility that one or more of the Voronoi cells could be pushed into in a different shell, thus comprising the shells of different sets of atoms. This composition would then be seen in differences in the number of atom types in a shell, average volumes of atom types in a shell, and in standard deviation calculations. We can further extend our research here by doing comparisons of shells created by power VT to shells created by additively weighted VT and analyzing their similarities and differences.

Our VMD results showed us the reach of molecules into different lipid and solvent shells, and how this reach helps construct these nearest neighbor relationships. Here, too, we can further extend our research by comparing power and additively weighted VT results in the weighted PDB to analyze any similarities or differences in the composition of the lipid or solvent shells. Is one VT method better than the other?

Coming into this project, we knew very little about the concepts of biophysics, much less the terminology used to describe it. The introduction into Voronoi tessellation theory, as well as creating our own algorithms to illustrate different types of tessellations and learning how to implement them in MATLAB and C++, truly enhanced our mathematical knowledge of this theory and helped show us how software might compute tessellations in similar manners. Given the chance to experiment with numerical methods on volumes and void spaces of proteins, we began to understand the computational applications that could be applied to these types of biophysical problems… and further research can hopefully make a dent in the many steps it will take to solve them.

REFERENCES

Cheng, Kelvin. “Presentation: Numerical Analysis 2013 VT Project.” 29 Jan, 2013.

Poirier, Michelle A. and Ross, Christopher A. “Protein aggregation and neurodegenerative disease.” Nature:Medicine. Jul 01 2004 Nature Publishing Group. Web. 5 May 2013.

Rycroft, Chris. “radical.cc – The radical Voronoi tessellation.” VORO++ Library README. Web. 25 Feb 2013. < http://math.lbl.gov/voro++/examples/radical/ >

“Voronoi Diagram.” Wolframalpha. Web. 5 May 2013. < http://www.wolframalpha.com/input/?i=Voronoi+diagram >