Advanced Physical Chemistry Final Portfolio

Brandon C. Allen*

Department of Chemistry, Monmouth College, Monmouth, Illinois

E-mail: ballen@monmouthcollege.edu

1 Introduction

This class was a conglomeration of various topics that would be explored in both Advanced Physical Chemistry and "Maker" courses. The class followed a general structure and explored a handful of different areas, allowing each student to explore various advanced topics in the process. This course required me to go outside of my comfort zone and taught me that being a scientist does not mean that I am incapable of being artistic and creative. I also got to learn a lot of great science along the way.

2 2-Dimensional → 3-Dimensional Transformation

This section of the course focused on transformations between the third and second dimensions. This was done by constructing three-dimensional objects from various "two dimensional" media and vice-versa. The ultimate application of the topics explored in this section was in showcasing all of these various things at a "STEAM" (Science, Technology, Engineering, Art, and Mathematics) fair in Bettendorf.

2.1 Paper Cutting

This section of the class focused on making three-dimensional objects from sheets of paper, as inspired by pop-up cards. The first object we constructed in this unit was a simple pop-out object using a piece of paper and strategically placed cuts with wavy scissors (Figure 1).

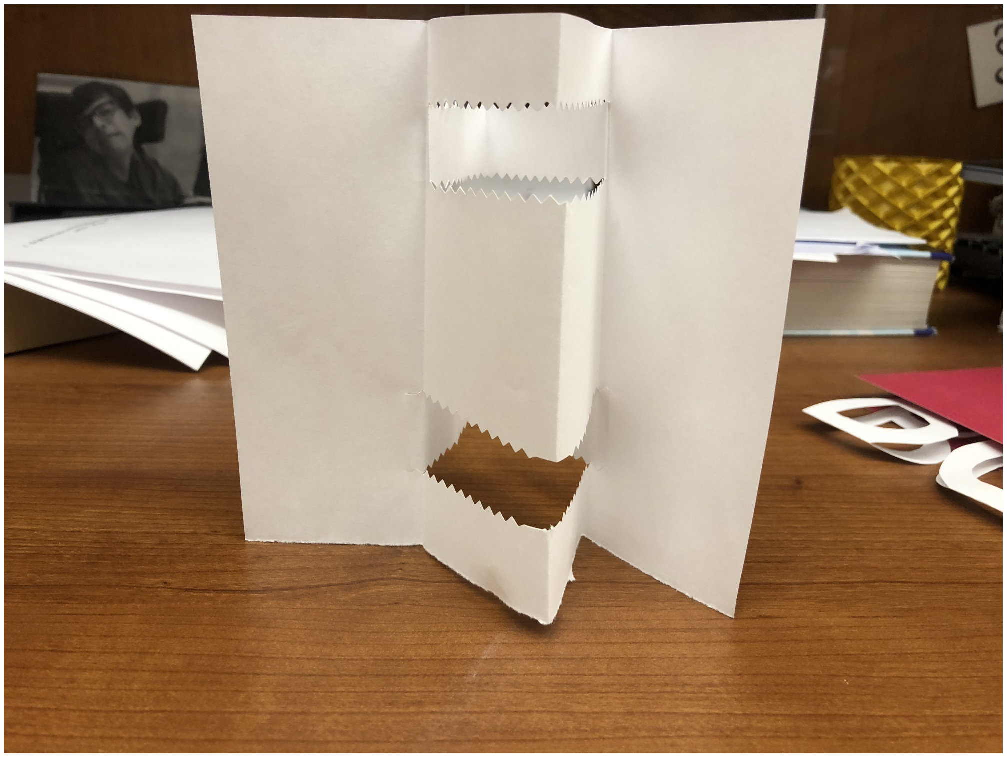


Figure 1: A generic 3-D pop-out object.

To expand on this idea, we then used a cutting mat and an Exacto knife to strategically cut our initials out of a piece of paper. This was then attached to a colored piece of construction paper, effectively constructing a pop-up card. Mine turned out pretty well as shown in Figure 2. Unfortunately, I left mine under a book, so it requires some added support to get the full "pop-up" effect. This can be fixed with a little bit of tape and gravity. Our exploration of paper cutting did not end here, as we continued to explore automation in the field of paper cutting by seeing how a Cricut machine cuts paper in a method similar to that of our laser cutter. This showed that with the creation of a proper file, we could get much more intricate

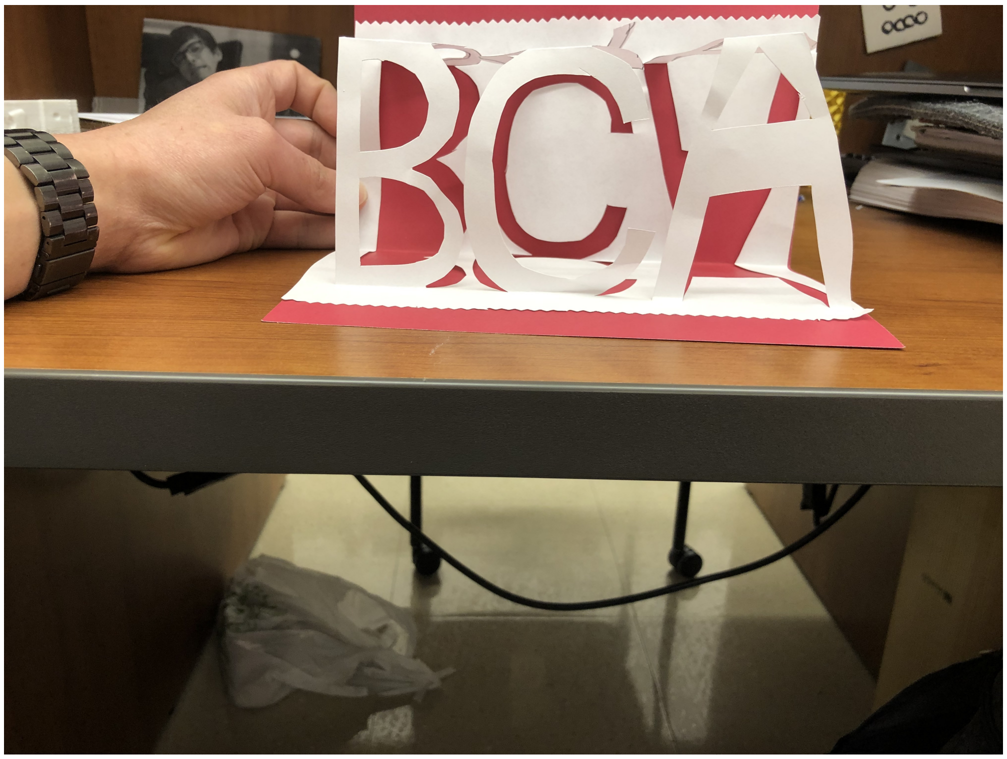


Figure 2: A pop-up card made depicting my initials (B.C.A.)

with the design of our pop-up cards. This is particularly appealing to someone like me, who is not very talented at making designs with my hands.

2.2 Living in the Third Dimension

To further explore the concepts of 3-D design, we investigated the use of 3-D printed molds to create clay figures. For this section, molds created and printed with the stereo-lithographic 3-D printer were used. These made awesome little figures such as the famous "mo‘ai" statues of Easter Island. (Also famously known as the figure that insults Ben Stiller as a "Dumb Dumb" in the *Night at the Museum* movie franchise.) To take things one step further, we explored transformation from the third dimension to the second dimension with light and a 3-D printed stereographic projection device. A stereographic projection is a map from the sphere to the plane. By the addition of a light source to this device, the two dimensional plane is reconstructed as shown in Figure 3.

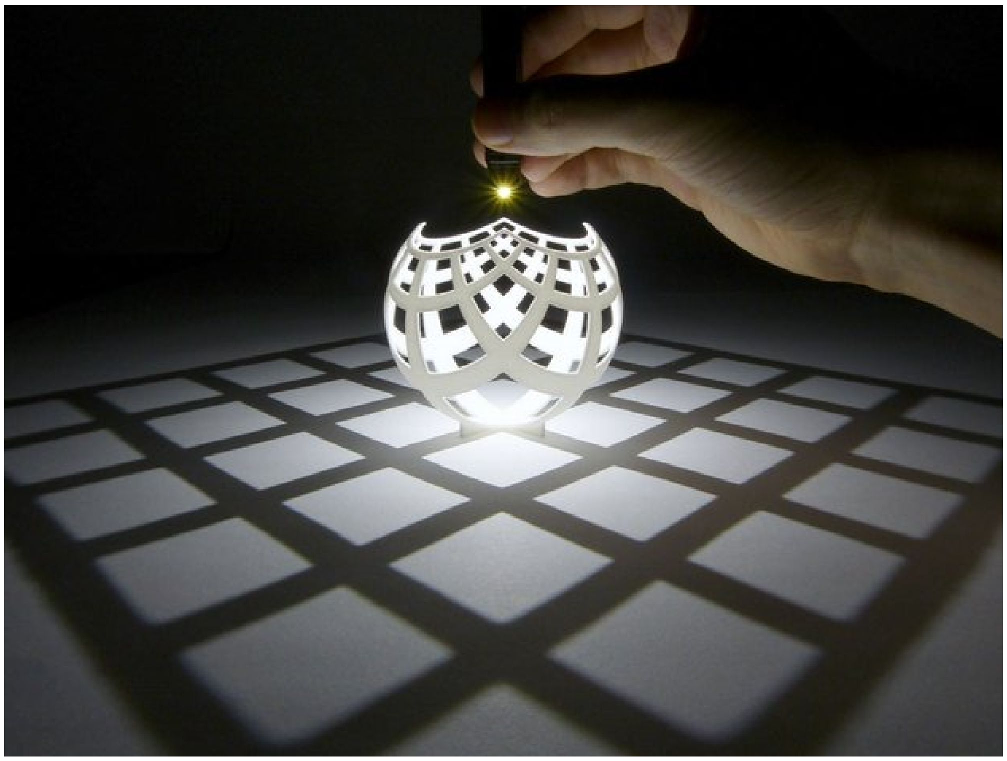


Figure 3: Stereographic projection, taken from: https://www.thingiverse.com/thing:202774

3 Adventures with the Raspberry Pi and Arduino

The next section of the course focused on the use of a Raspberry Pi computer (Figure 4) and Arduino microcontrollers, fundamental components in any "Maker's" toolbox.

3.1 Raspberry Pi Set-Up

The first step in this process was to install the "NOOBS" (New Out Of the Box Software) operating system onto the micro SD card storage system for the Raspberry Pi. This was down by downloading NOOBS from the Raspberry Pi website and then extracting the contents to the micro SD card. Once this was inserted into the Pi, the computer was put into an acrylic case and connected to a computer mouse, monitor, and keyboard. Then a straightforward installation process (https://www.raspberrypi.org/help/noobs-setup/2/) was followed to set-up the operating system for the Pi. Once this was completed, the Pi was updated by entering the following commands into the Terminal on the Pi:

```
sudo apt-get update
sudo apt-get upgrade
```

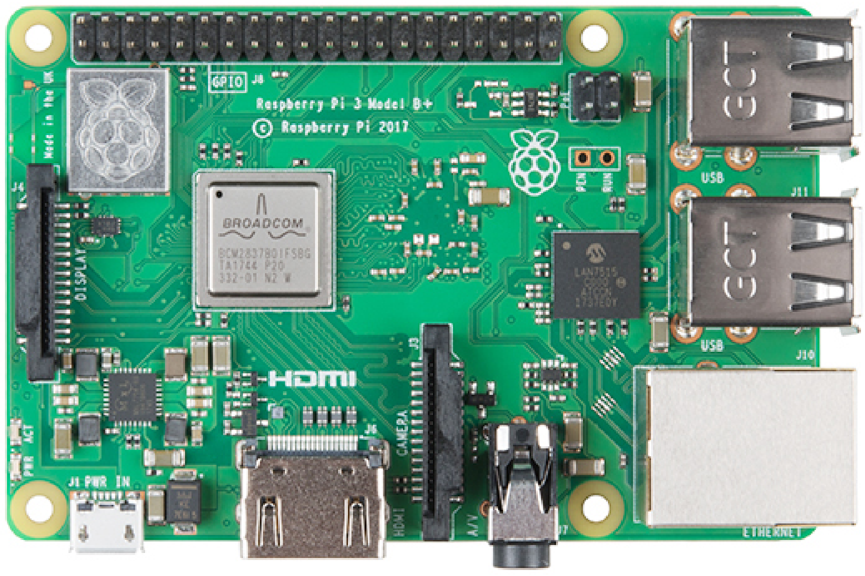


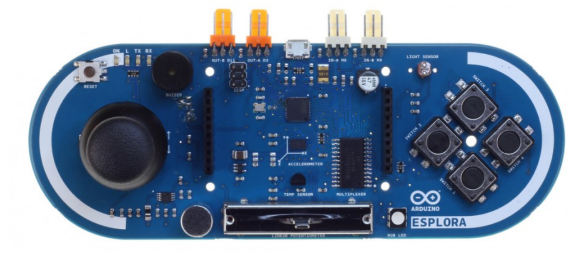
Figure 4: Picture of the Raspberry Pi 3 B+ Computer

Next, the Arduino IDE (Integrated Development Environment) was installed onto the Pi to allow for ease of the development of Arduino code and direct communication with the Pi. This was done in the Terminal via the following command:

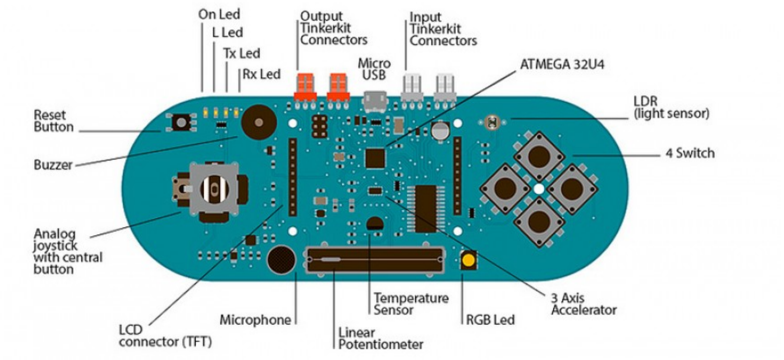
```
sudo apt-get install arduino
```

3.1.1 Putting the Raspbery Pi to Use - Communication with an Arduino Esplora

An Arduino Esplora (Figure 5) was connected to the Pi via USB serial connection. This allowed the Pi to both power and communicate with the Esplora such that we could flash code to it through the Arduino IDE. This was used to explore the Arduino programming language and how it controls the various on-board sensors that the Esplora comes with. After



(a) Image of the Arduino Esplora Board.



(b) Arduino Esplora Board Schematic

Figure 5: Arduino Esplora Board and Labeled Diagram

learning how to communicate with the various on-board sensors of the Esplora by using the examples, we then had to design an example project of our own. For this, I decided I would use the joystick of the Esplora to independently control two servo motors. While figuring out how to wire up the servos, I came across an interesting diagram (shown in Figure 6) that explained what the various pinouts on the board can be used for.

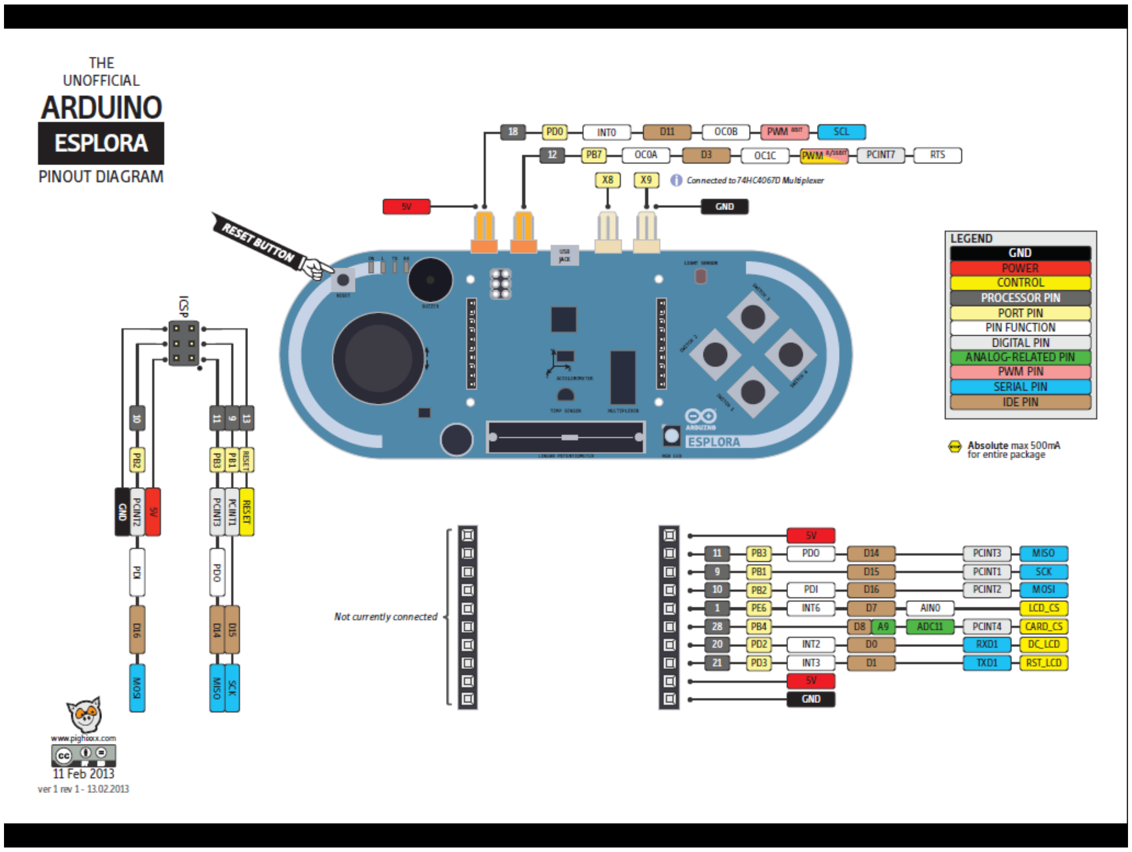


Figure 6: Arduino Esplora Pinout Diagram

This provided me with the information I needed to wire up two servo motors and successfully communicate with them independently, such that moving the joystick up/down would move the first servo to the left/right and moving the joystick right/left would move the second servo to the left/right. I hope to make a cool application for this project someday, such as a creepy moving head.

This was done with the following code:

```
#include <Esplora.h> //Libraries for Esplora
#include <Servo.h> //Libraries for Servo Communication
Servo x; //Servo 1
Servo y; //Servo 2
void setup() {
  x.attach(14); //Change both numbers according to location of
     servo's pin
  y.attach(0);
  Serial.begin(9600); //Begin serial input
}

void loop() {
  int valx = Esplora.readJoystickX(); //Read joystick X value
  int valy = Esplora.readJoystickY(); //Read joystick Y value
  Serial.println(valx, valy); //Print values to serial monitor
  int valx1 = map(valx, -512, 512, 0, 180); //Map both joysticks
     value to servo acceptable values
  int valy1 = map(valy, -512, 512, 0, 180);
  x.write(valx1); //Write to both servos
  y.write(valy1);
  delay(200); //Delay. Prevents overwhelming the arduino
}
```

3.1.2 Poseidon Syringe Pump

In addition to making a "cute" twitchy servo motor project by combining the power of an Arduino microcontroller and the Raspberry Pi computer, I was able to put together a series of syringe pumps (shown in Figure 7). This was done according to the instructions provided by the Poseidon Syringe Pump project from the Pachter Lab at Caltech. This project involves the control of three stepper motor through the use of a CNC Shield that connects directly to an Arduino Uno and is controlled by software running on a Raspberry Pi.

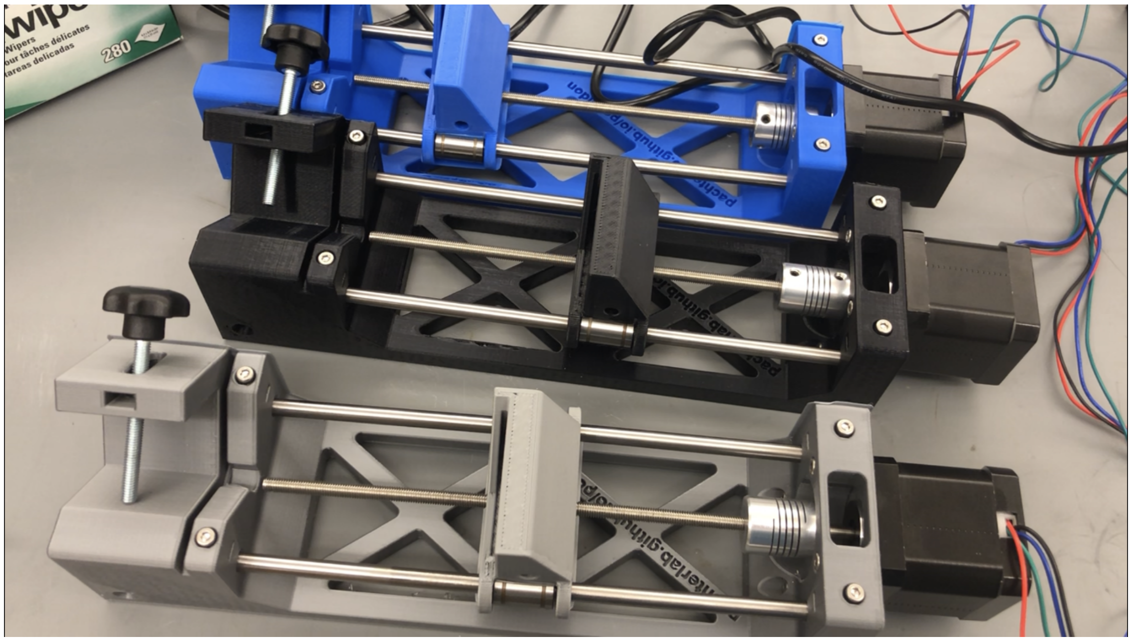


Figure 7: Picture of the Syringe Pumps

More information about this project can be found at:

```
http://esr.monmsci.net/wiki/index.php/Poseidon_Syringe_Pump
```

4 Computational Chemistry and Molecular Visualization

The next focus of the course involved computational chemistry and molecular visualization programs. In this subject, we learned about various software packages that can be used for doing powerful calculations that can enhance the quality of work done in the research lab.

4.1 WebMO Pro

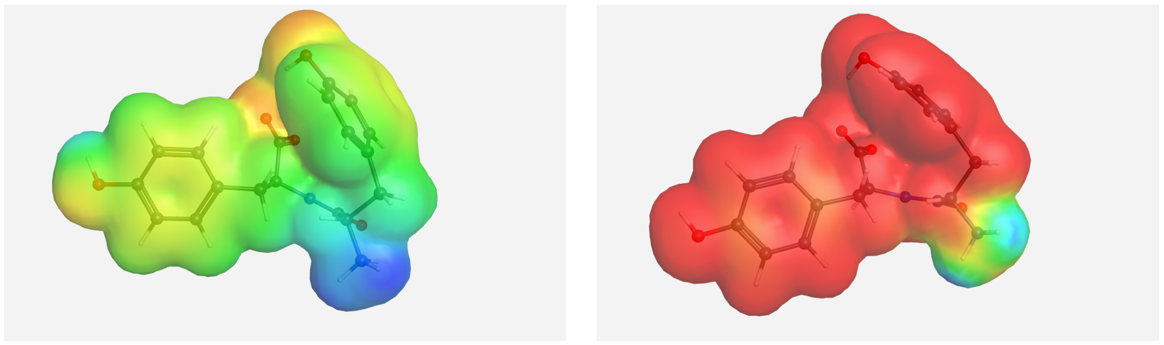
We used Gaussian via access to a WebMO Pro server to discuss and perform various calculations.

4.1.1 Molecular Energy

A Molecular Energy calculation computes the energy and electronic properties (dipole moment, partial charges, bond orders, etc.) of the whatever atomic/molecular structure is given as input.

4.1.2 Molecular Orbitals

A Molecular Orbitals calculation computes the molecular orbitals of the input, from which the electron density, electrostatic potential, and frontier orbital densities can be calculated. The output of this calculation can be visualized as the colorful electrostatic potential maps given in various textbooks, shown in Figure 8.



(a) Electrostatic Potential Map (b) Radical Frontier Density

Figure 8: Various outputs from a Molecular Orbital calculation on a di-Tyrosine peptide

4.1.3 Geometry Optimization

A Geometry Optimization calculation alters the geometric configuration of the input molecule to find the nearest energy minimum. It then reports the resulting energy, electronic properties, and geometry as output. The optimized geometry is the nearest local minimum, which is not necessarily the global minimum. (Insert philosophical comment about avoiding being trapped in the local minima of life here.) A simple geometry optimization of glycine was done to illustrate this calculation type, as depicted in Figure 9.

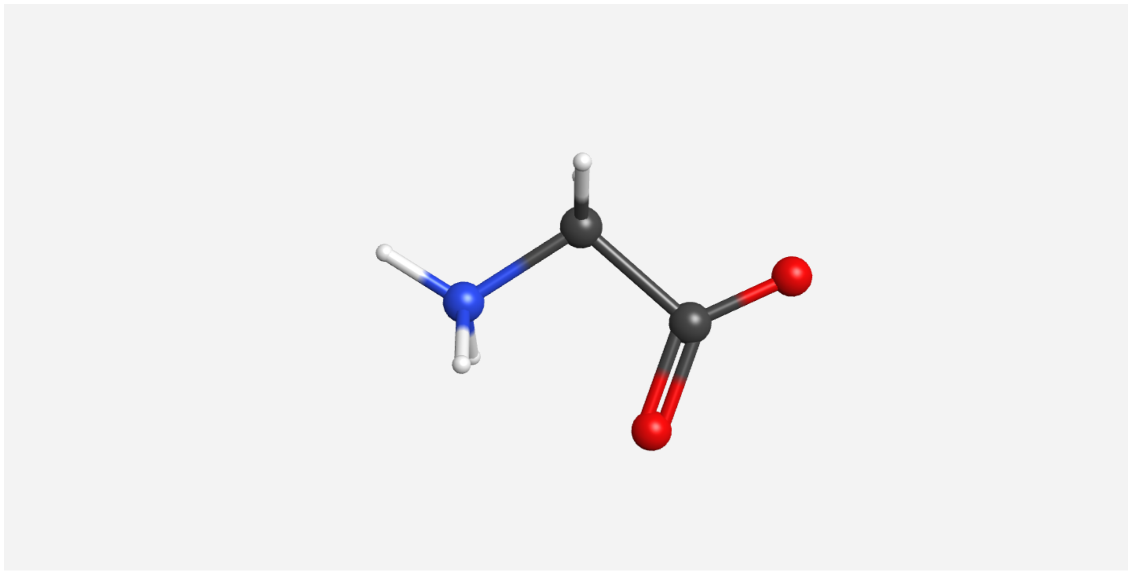


Figure 9: Example Output of a Geometry Optimization Calculation on Glycine Amino Acid

4.1.4 Vibrational Frequencies

A Vibrational Frequencies calculation finds the normal vibrational frequencies, intensities, and modes of a molecule. This requires a geometry optimization to be run prior to running this job.

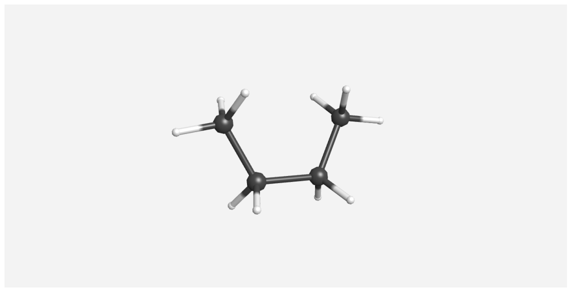
4.2 Coordinate Scan

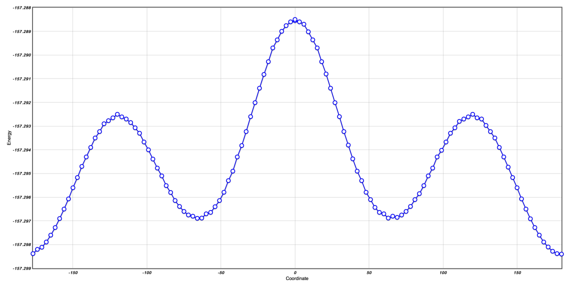
A Coordinate Scan calculation allows for iterative change of a specific variable/coordinate (bond length, bond angle, or dihedral angle) and computes the energy at each point. The remaining coordinates can be all optimized (relaxed scan), all fixed (rigid scan), or a combination of both.

Prior to selecting this calculation type, the coordinates to be scanned must be defined using the Adjust Tool of the WebMO Editor. Alternatively, the Z-Matrix Editor can used to define the coordinate to be scanned (S), and whether the remaining coordinates should be optimized (O) or fixed (F).

The resulting energy values can be visualized on the Job Results page or exported in spreadsheet format.

To test this calculation out, a simple butane was used, in which a coordinate scan of the dihedral angle between the external carbons was calculated (Figure 10).


(a) Optimized Butane geometry.



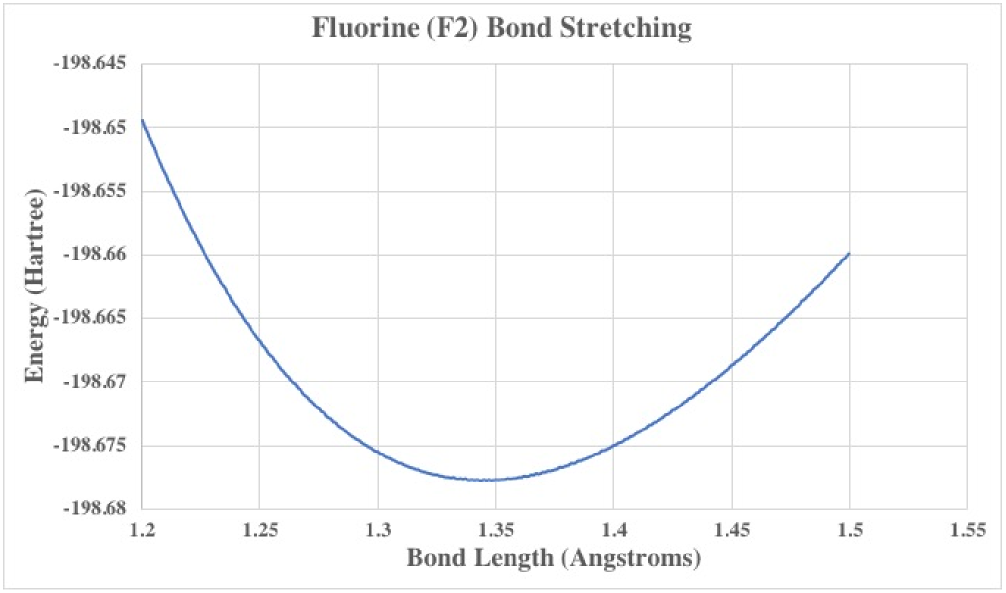
(b) Output of the coordinate scan.

Figure 10: Molecular input for the coordinate scan and the energy plot resulting from the calculation.

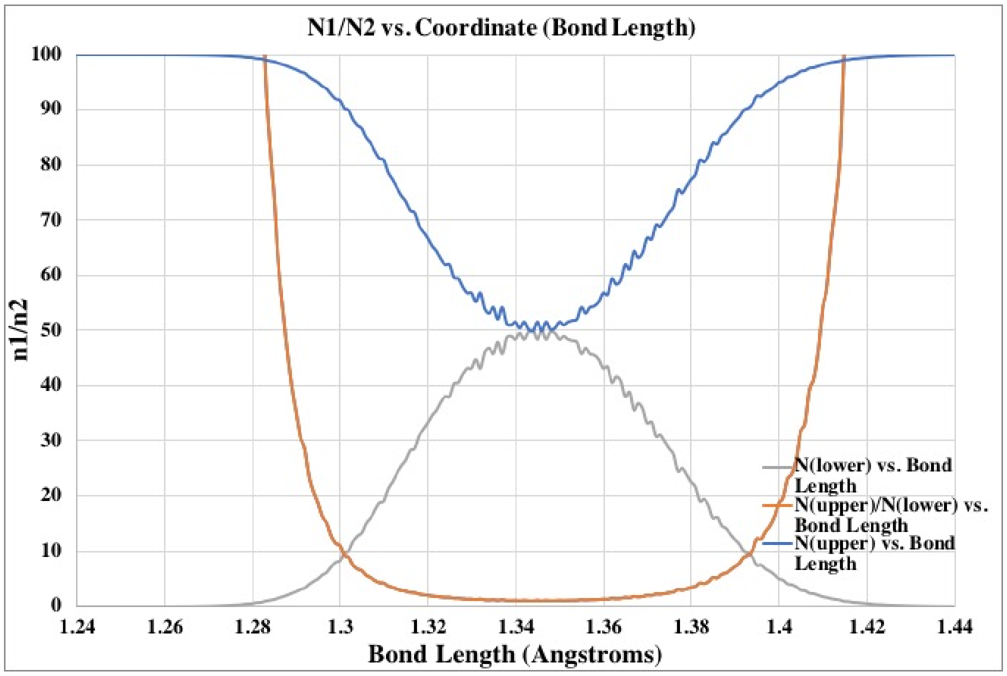
4.2.1 Coordinate Scan of Bond Length in F_2

The first thing we did with WebMO Pro was investigate a simple calculation of geometry optimization in a homodiatomic molecule. I chose F_2 and decided to use a coordinate scan for my calculation of the Molecular Energy, in which I changed the length of the bond between the two Fluorine atoms while calculating the Molecular Energy. This provided some interesting data for a discussion of the Boltzmann Distribution.

This shows that there is a "perfect" bond length at which the repulsive and attractive forces are "balanced" and the lowest energy state can be achieved. This directly relates to discussion of the Boltzmann Distribution, which describes the population of relative energy states. To illustrate the Boltzmann Distribution, the populations of the upper and lower energy states were compared (Figure 11).



(a) WebMO Pro Output from calculation of Molecular Energy while doing a coordinate scan of the bond length.

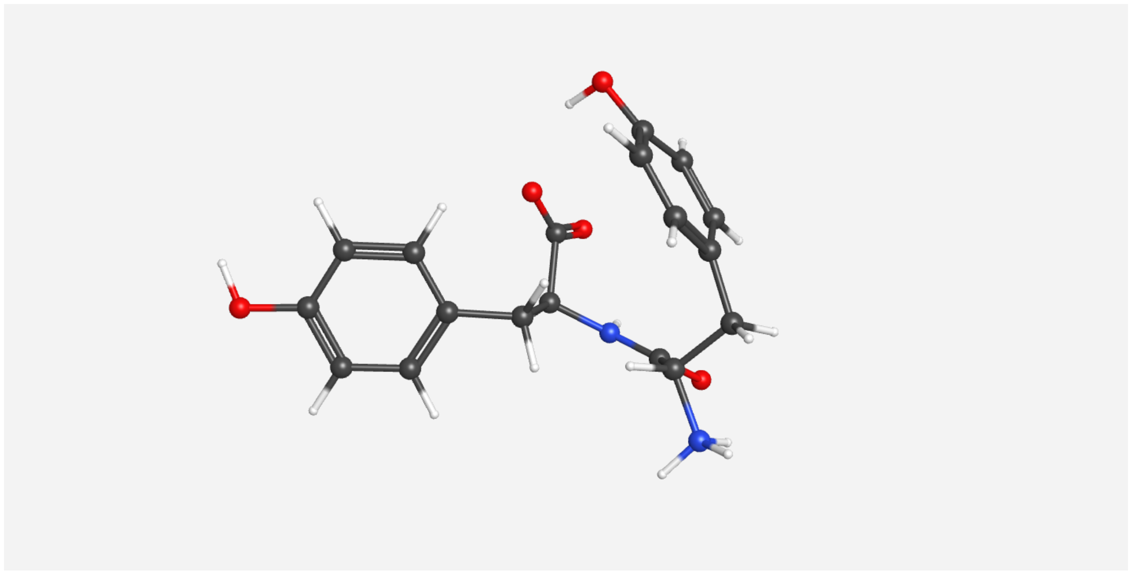


(b) Plot Showing the Boltzmann Distribution for the Bond Stretching Coordinate Scan

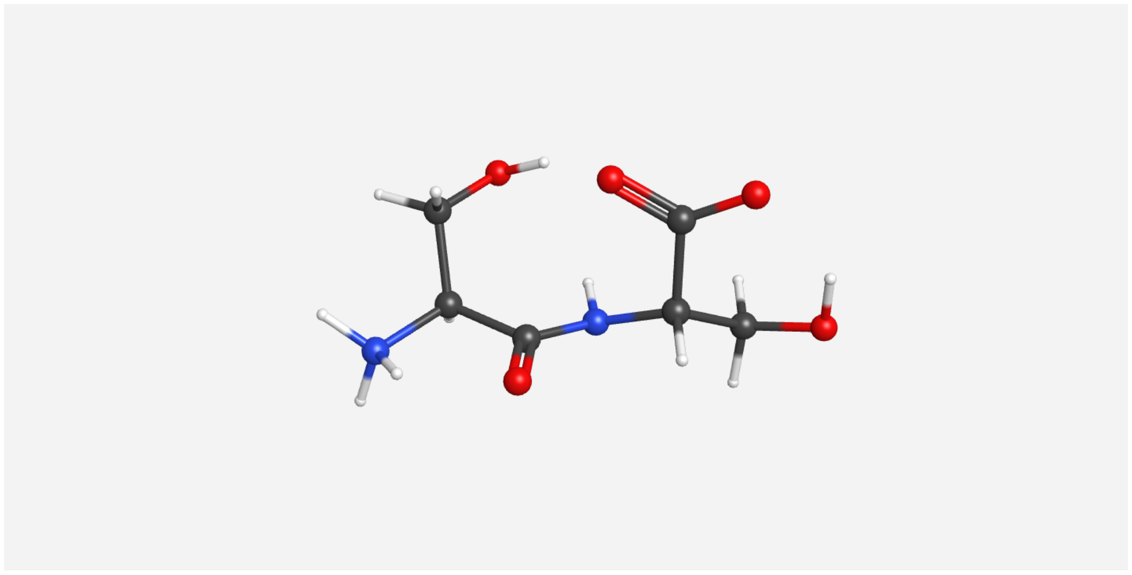
Figure 11: Resulting Data from Coordinate Scan of Bond Stretching in F_2 molecule.

4.2.2 Amino Acid and Peptide Calculations

A series of dipeptide calculations were done on dityrosine and diserine peptides. First, geometry optimizations were done to give bond lengths of key bonds for comparison to a paper we read (Figure 12).



(a) Optimized dityrosine peptide geometry.



(b) Optimized diserine peptide geometry.

Figure 12: Geometry Optimization output for calculations done on two dipeptides.

The geometry optimization was done to see what these dipeptides looked like as the chemical nature of the R-group of the generic amino acids were changed.

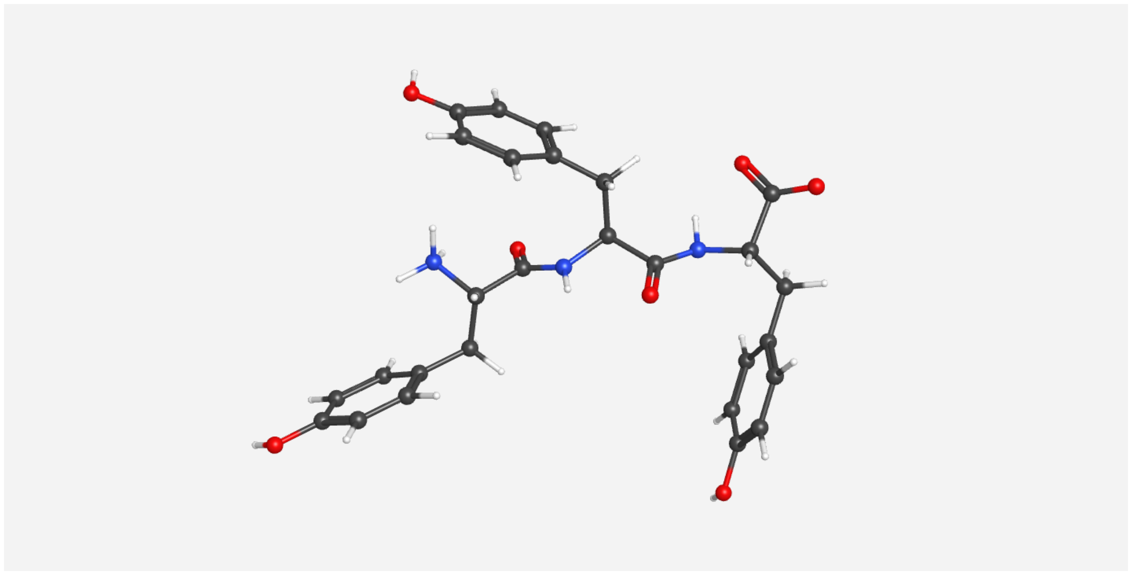


Figure 13: Optimized trityrosine peptide geometry.

Once this was completed, a geometry optimization of a tripeptide was done. The tripeptide that I chose to use was trityrosine (Figure 13). This was done in preparation for a coordinate scan of the dihedral angles between R-groups in the tripeptide.

4.3 I-TASSER

To expand on our exploration of computational chemistry resources and the various peptide calculations done in Gaussian with WebMO Pro, our conversation shifted to protein folding and the number one algorithm in the field - I-TASSER. To completely illustrate the power of this software, we submitted various amino acid sequences to I-TASSER. The sequence that I chose to work with was the Amyloid-β peptide. It simply took an input of the amino acid sequence and provided robust output, which included the 5 most likely structures (Figure 14). To test the capabilities of I-TASSER, we then changed one of the amino acids in our original sequence and resubmitted the job to see how the structure changed. This is referred to as site-directed mutagenesis in a laboratory context. The results are shown in Figure 15.

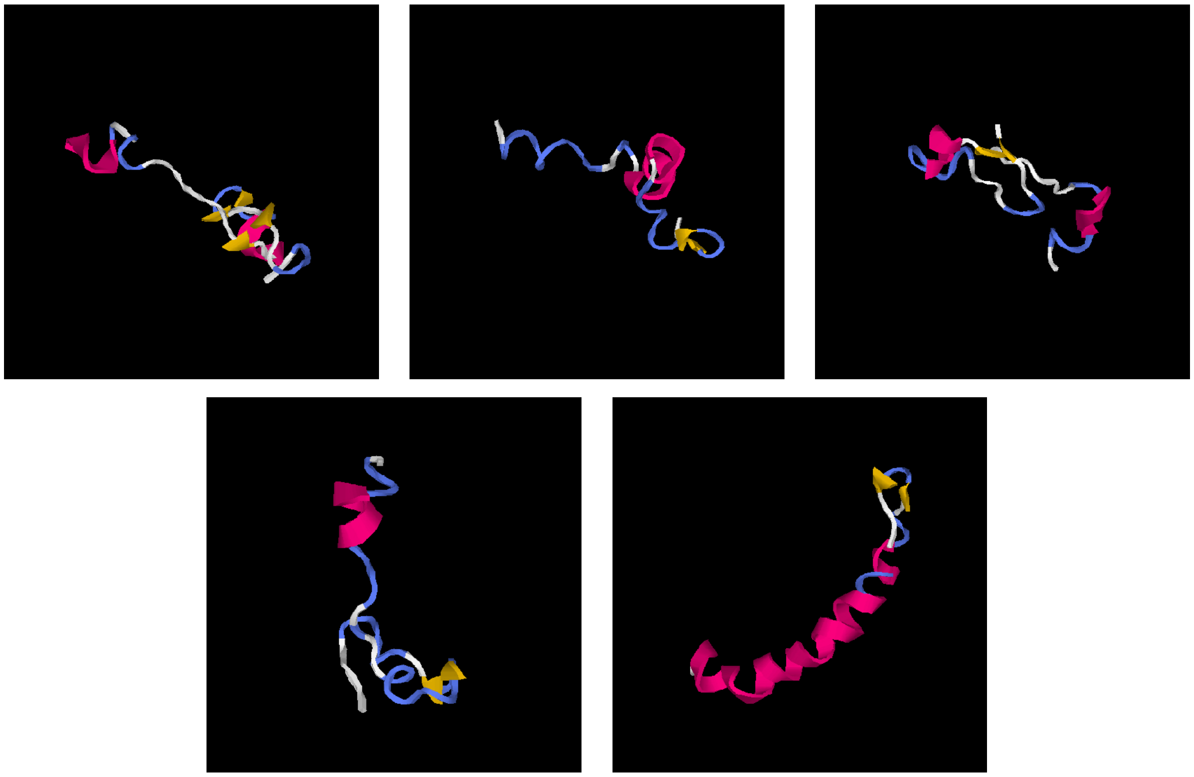


Figure 14: Example I-TASSER Protein Folding Calculation Output for Amyloid-β (1-40) Peptide

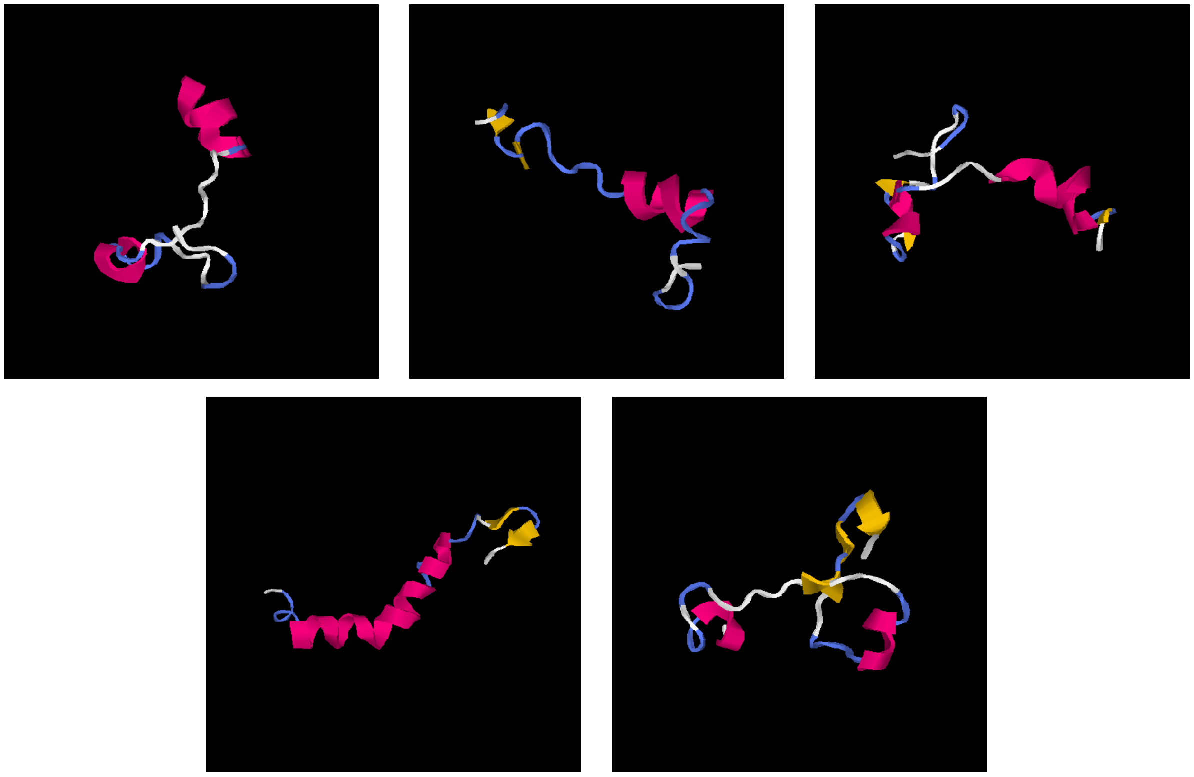


Figure 15: Example I-TASSER Protein Folding Calculation Output for Mutant Amyloid-β (1-40) Peptide

4.4 Molecular Visualization

While doing our work with protein folding and peptide calculations, it became clear that a molecular visualization tool is a necessity when working with computational software. This led us to explore two common visualization software packages, PyMOL and JMOL.

4.4.1 PyMOL

PyMOL is a powerful molecular visualization tool that provides a Python interface on top of a user-friendly GUI. This is a useful tool for the visualization of .pdb (Protein Data Bank) files, as well as the output files of many popular Molecular Dynamics simulations. Some of the various commands that I came across that may be useful for introductory users:

```
fetch 1ubq # fetches 1ubq from the protein databank
load /path/to/file.pdb # command used for loading a file on computer
```

```
show sticks # shows the sticks representation of the protein
show lines # shows the lines representation of the protein
show cartoon # shows the cartoon representation of the protein
# hide sticks/lines/cartoon will allow you to hide
# whichever representation of the protein you want
save filename.pdb # saves the file
# More shortcuts can be found at:
# https://pymolwiki.org/images/7/77/PymolRef.pdf
```

With the use of a molecular visualization program like PyMOL, I was able to load the I-Tasser output, making it easier to explore the structures that it calculated (Figure 16). I

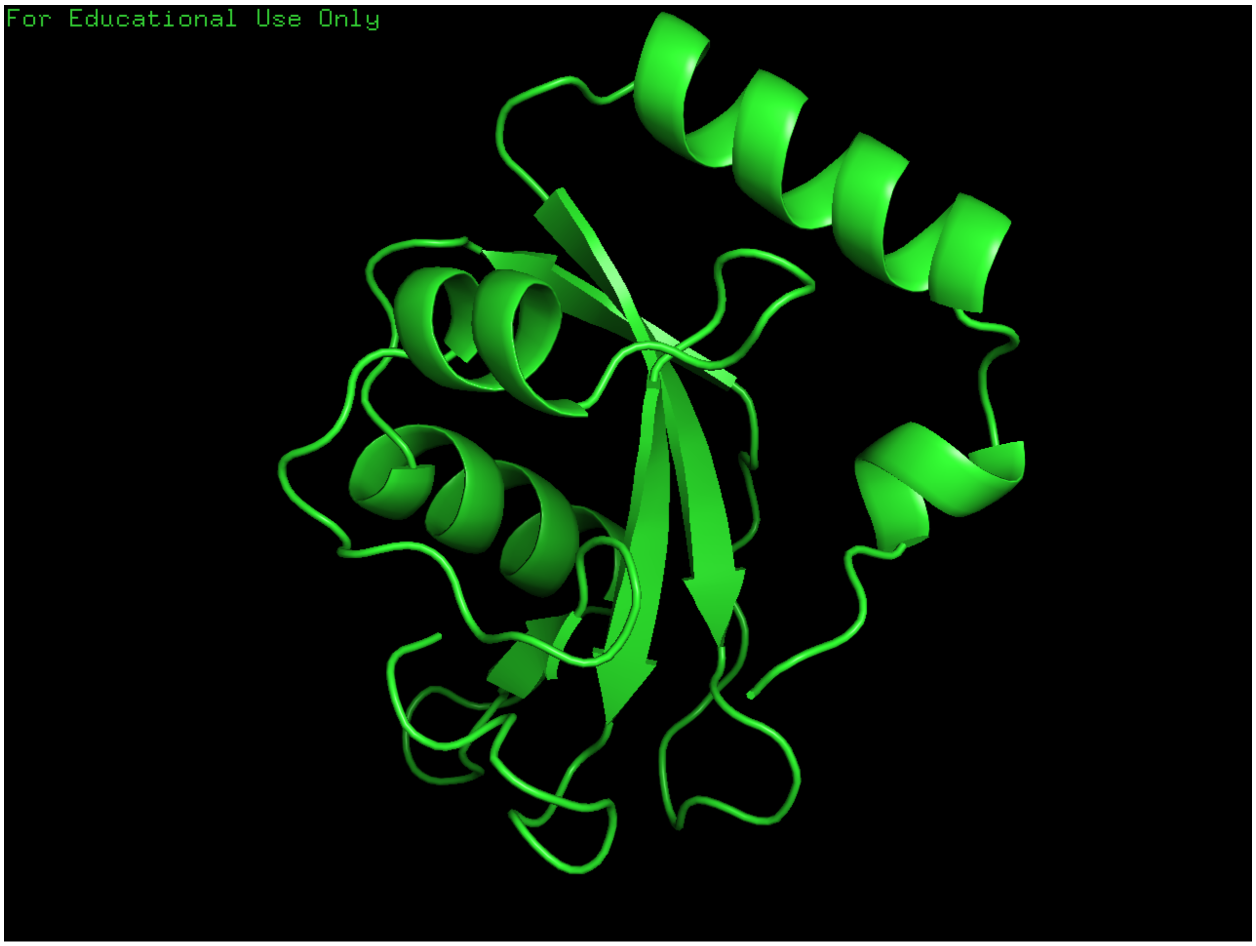
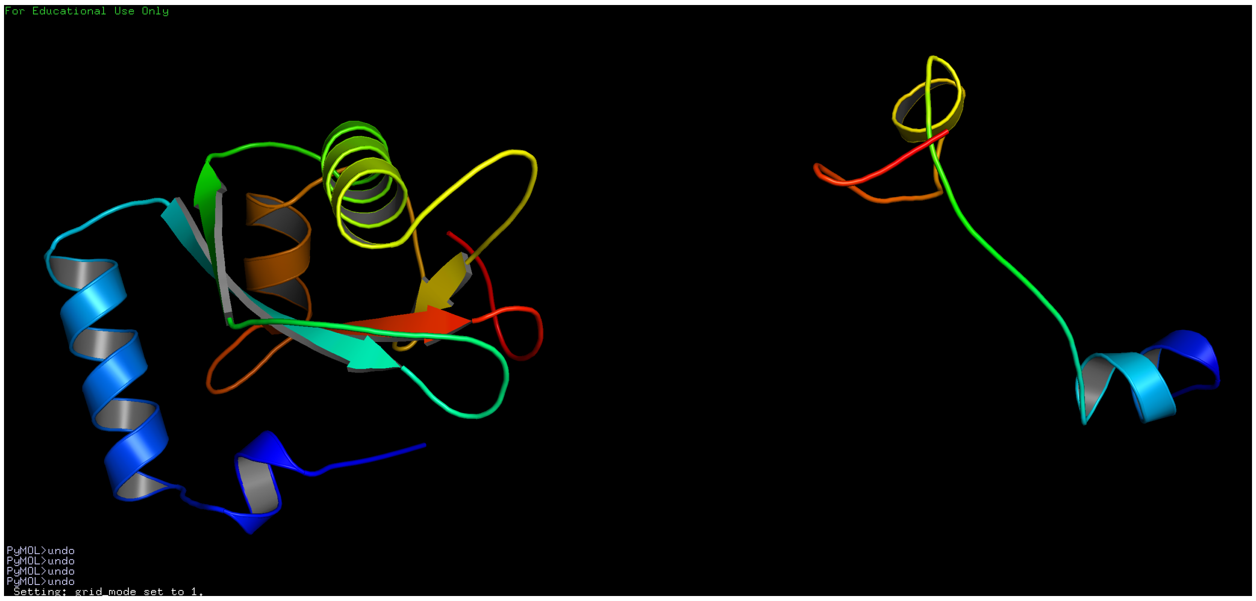


Figure 16: I-TASSER Calculation Output for Mutant Amyloid-β (1-40) Peptide in PyMOL

was then able to take this model and compare it side-by-side to the most-likely structure given by I-TASSER after I changed one of the amino acids in the sequence (Figure 17). This program serves as a very simple and user-friendly software package for viewing the 3D



```
PyMOL>undo
PyMOL>undo
PyMOL>undo
PyMOL>undo
 Setting: grid_mode set to 1.
```

Figure 17: I-TASSER Calculation Output for Mutant Amyloid-β (1-40) Peptide and Mutant in PyMOL

structures of large molecules and also provides an environment to do some post-processing calculations. This served as a source of inspiration for a later project for Scholar's Day.

4.5 Molecular Dynamics Simulations with Gromacs

I chose to explore the subject of computational chemistry a bit further by looking into Molecular Dynamics simulations. Molecular dynamics (MD) models properties of a system of interacting particles by iteratively calculating the interactions between the particles and integrating their equations of motion, resulting in a trace of a discrete phase-space trajectory. This is done by combining statistical mechanics and kinetic theory, allowing for the calculation of microscopic properties of the system. The interactions are calculated over time-steps. To explore this further, I chose to use a common MD software called Gromacs. This is a powerful MD simulation software package that is run through the terminal.

4.5.1 Lysozyme in Water

To further explore the process of constructing and running an MD simulation, I followed a tutorial for simulation of lysozyme in water. Lysozyme is an enzyme that has the ability to kill bacteria and it is present in various things including tears, saliva, and egg whites.

The first step of this tutorial was to set-up Gromacs on my computer. This is described in detail elsewhere. Once Gromacs was set-up I had to download the structure of the lysozyme protein from the Protein Data Bank. The structure used was 1LYD.pdb To ensure that this was a good structure for the purposes of running a simulation, I then viewed the protein in PyMOL to ensure that there wasn't an issue with the resolution or actual composition of this structure.

To prepare the structure for simulation, I had to create a topology. This is done easily with a simple command in Gromacs.

```
pdb2gmx -f 1LYD.pdb -water tip3p
```

The program then asks for a forcefield to use for the calculation. This forcefield is a predefined function that subjects the molecules in the simulation to a specific series of forces, resulting in the changes that are used for the final simulation product. The forcefield that I used was "OPLS-AA/L".

The next step in the simulation was to solvate the protein with water. Prior to adding the water molecules, a box to constrain the molecules within must be defined. The size of the box has a significant impact on how computationally intensive the simulation is. The box used for my simulation was 0.5 nanometers, and the volume of the box was minimized by use of a rhombic dodecahedron box shape. This was defined with the following command:

```
editconf -f conf.gro -bt dodecahedron -d 0.5 -0 box.gro
```

Once this was done, the water molecules were added to the box. The lysozyme system requires approximately 6000 water molecules, which increases the atoms in the simulation to over 20,000. A pre-equilibrated water system was not used. The command used to solvate the box is:

```
genbox -cp box.gro -cs spc216.gro -p topol.top -o solvated.gro
```

The solvent coordinates were taken from an SPC water system, and the -p flag added the new water molecules to the pre-existing topology. To ensure that everything so far had gone according to plan, the system had to be viewed with PyMOL. This was done by using the following command:

```
trjconv -s solvated.gro -f solvated.gro -o solvated_compact.pdb /
-pbc atom -ur compact
```

The "/" is included to indicate that these should all be on the same line. Next, an energy minimization was run. This is done in order to remove the forces generated by the hydrogens and broken hydrogen bond network in water. These interactions had to be defined in a specified parameter file, named **em.mdp**.

Next, the Gromacs preprocessing program **grompp** was used in order to collect the parameters, topology, and coordinates into a single run input file from which the simulation can be started. This is done with the following command:

```
gromp -f em.mdp -p topol.top -c solvated.gro -o em.tpr
```

This resulting tpr file was run through a simulation with the following command:

```
mdrun -v -deffnm em
```

The **-deffnm** is a shortcut that uses "em" as the base filename for all options, but with different extensions. This took about 20 minutes to run.

Next, the water around the protein needed to be equilibrated. This is done to avoid any unnecessary distortion of the protein during the MD simulation. This essentially restrains the heavy protein atoms to their starting positions while the water surrounding the structure is relaxed. This was done for 5 picoseconds. Bonds were constrained to enable 2 femtosecond time steps. The settings used were stored in a file named **pr.mdp**:

```
——pr.mdp——
define            = -DPOSRES
integrator        = md
nsteps            = 2500
dt                = 0.002
nstlist           = 10
rlist             = 1.0
coulombtype       = pme
rcoulomb          = 1.0
cutoff-scheme     = verlet
vdw-type          = cut-off
rvdw              = 1.0
tcoupl            = v-rescale
tc-grps           = protein  non-protein
tau-t             = 0.1  0.1
ref-t             = 298  298
Pcoupl            = Berendsen
tau-p             = 1.0
compressibility   = 1e-5 1e-5 1e-5 0 0 0
ref-p             = 1.0
refcoord-scaling= all
nstenergy         = 100
constraints       = all-bonds
```

The tutorial stated that a small protein such as lysozyme should be sufficiently equilibrated with the surrounding water in 100 ps, but a large membrane system (such as that composed of phospholipids) can require several nanoseconds to full relax. To know for certain that the equilibration was successful, we had to watch the potential energy and extend the equilibration until it converged. The equilibration was run via:

```
grompp -f pr.mdp -p topol.top -c em.gro -o pr.tpr
```

This was then run with:

```
mdrun -v -deffnm pr
```

Next, the production simulation was run. There are only small differences between the equilibration and production simulation. The main difference is that the position restraints and pressure coupling are turned off (ex: every 50 steps), the rate at which the output coordinates are written is determined, and this is a much longer simulation. This depends on the system being studied. For decent sampling, the simulation should be at least 10 times longer than whatever the process being studied is. The simulation was run with the following .mdp file. This contains output data from a 10 nanosecond simulation of the system (with 5 million steps, this took about 8 hours to complete on my personal machine). The .mdp file has the following structure:

```
—— run.mdp ——
integrator          = md
nsteps              = 5000
dt                  = 0.002
nstlist             = 10
rlist               = 1.0
coulombtype         = pme
rcoulombtype        = 1.0
cutoff-scheme       = verlet
vdw-type            = cut-off
rvdw                = 1.0
tcoupl              = v-rescale
tc-grps             = protein non-protein
```

```
tau-t                = 0.1  0.1
ref-t                = 298  298
nstxtcout            = 1000
nstenergy            = 1000
constraints          = all-bonds
```

The production run was performed with:

```
grompp -f run.mdp -p topol.top -c pr.gro -o run.tpr
mdrun -v -deffnm run
```

Finally, the analysis of the simulation was done by visualizing the trajectory file with PyMOL. This was done with the following command to create a PDB format movie:

```
trjconv -s run.gro -f run.xtc -e 2500.0 -o movie.pdb
```

This was easily loaded into PyMOL and saved as a quicktime file to share with others, as shown in Figure 18!

5 Scientific Discussion

The next portion of the course focused on scientific discussion. Although being able to produce quality scientific results is important in our field, these results are effectively rendered useless if we can not effectively communicate them to other people. To look deeper into this, we reviewed a manuscript, discussed the process and potential issues encountered in the scientific writing process, and discussed an article about peptides. During the review of the manuscript, we discussed strategies for efficiently analyzing/assessing a paper. This also led to a discussion of the process that is followed in the field for publication of a manuscript. During this discussion, we read/reviewed a manuscript that was submitted for publication and discussed whether we felt it was worthy of publication. This taught me a lot about

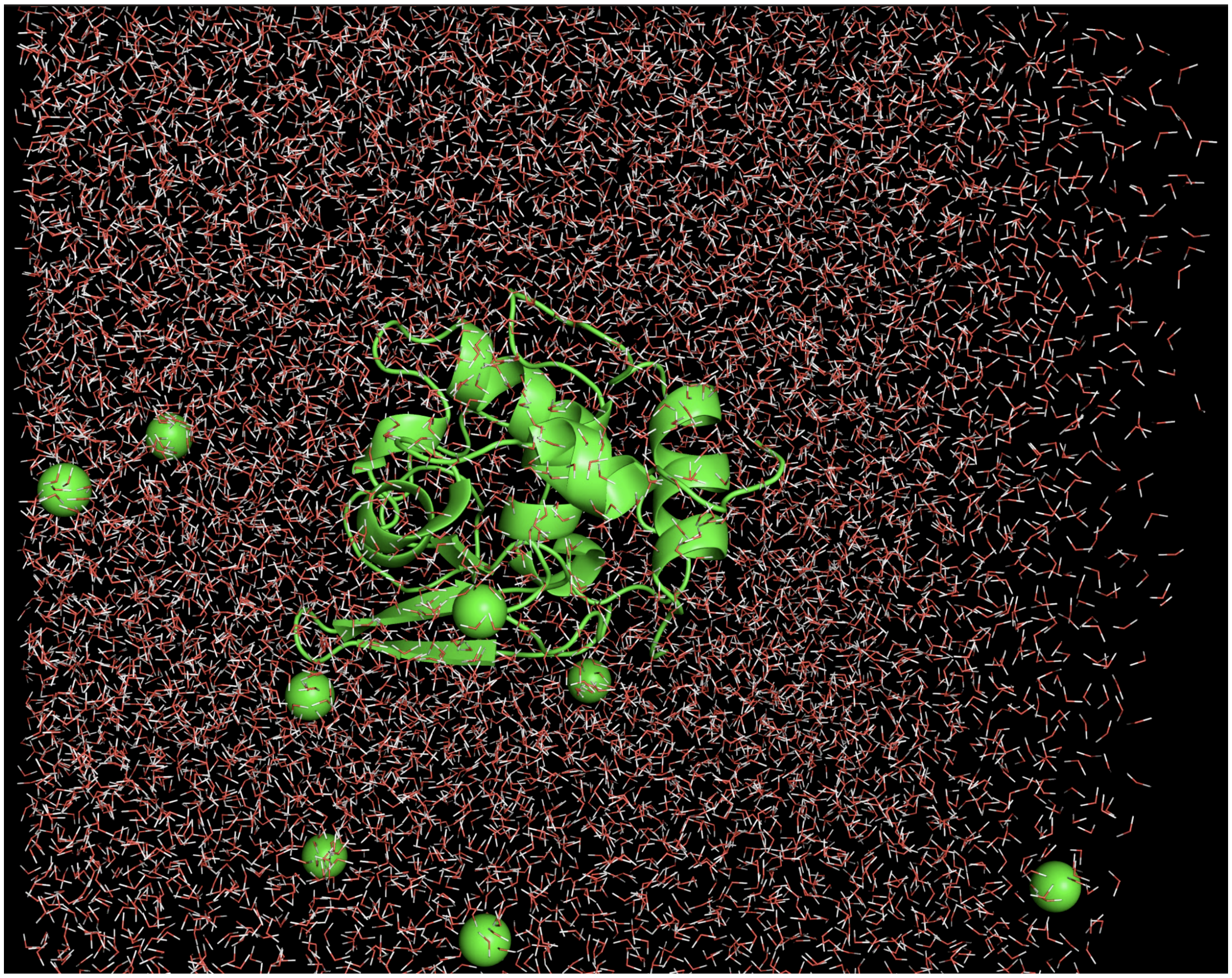


Figure 18: Lysozyme MD Simulation running in PyMOL

the considerations to make while putting a manuscript together. During this portion of the course, we also read an article that was featured in Chemical & Engineering News, titled “Exploring allostery’s advantages”. This was an interesting article and opened a discussion of the various roles that a chemist can take in an industrial/business setting. It also revealed the importance of computational software, such as that written by Schrodinger (the company that developed PyMOL), in the pharmaceutical industry and drug discovery process. During our discussion of these various scientific resources, we explored peptides further and read “Discovering Enzymes” by David Dressler. This was an excellent resource for the protein folding and peptide calculations that were run in Gaussian through WebMO Pro, as mentioned previously.

6 Maker Explorations

The underlying goal of this class was to develop a model for a "maker" course. So, the final portion of the class shifted to utilizing the resources that we have here at the CSB for the creation of things. The first project that was done in this section was cutting, sanding, and processing wood for the construction of flag stands. These flag stands are used to create a more "internationally welcoming" environment on our campus by allowing professors to showcase the various countries that they have visited with small flags from each of those countries. To facilitate this project, we had to construct various flag stands (varying in size and carrying capacity) by using tools in the workshop. This involved cutting pieces of wood down to size, sanding them stylistically to look nice, and then staining and sealing the final product. We produced a lot of flag stands in the time that was spent in the workshop, and I have seen a handful of them floating around various professors' offices.

In addition to the flag stands, a "wall of monitors" set-up was constructed to showcase at Scholar's Day. This was constructed with various pieces of wood on a cart. This required consideration of the proper method for powering 9 monitors, 9 Raspberry Pis, and the best way to mount these things such that they would not be destroyed in the process of transferring them over to the Huff Athletic center. I was less involved in the creation of the wall of monitors than I was with the software aspect. On the wall of monitors, we wanted to display our calculation output from the protein folding jobs run with I-TASSER. In order to do this, we had to figure out a way to run a molecular visualization program on a minimal capacity computer - the Raspberry Pi. Since I had previously worked with PyMOL, I focused on finding a way to get PyMOL to run on the Pi. This involved spending a decent amount of time figuring out how to get the software on the Pi. Eventually, I was able to install the software via the following command:

```
sudo apt-get install pymol
```

Once this was installed, we were able to load our file into PyMOL on the Pi, but quickly realized why it was so difficult to get the software onto the Pi in the first place - it really is not meant to run on the Pi and has definitely not been optimized for this configuration. While trying to get the structure to “rock” back and forth, the software began strobing on and off, like it had to think really hard about each step of the “rocking” process. After spending some time trying to troubleshoot this issue, we abandoned the idea of using PyMOL for our display. This led to the use of JMOL, which is similar to PyMOL in functionality and much better configured for the resources available in the Raspberry Pi computer. This was successfully used to construct an awesome “wall of monitors” display for Scholar’s Day.

To expand on the “maker” topic of the course, we explored various innovations in the field of automation. This was done in the hope of learning more about an automatic pipetting system created by a company called OpenTrons. We had the opportunity to video chat with a sales representative from the company and learned a lot about the capabilities of this automation system. In addition to this, I had the opportunity to attend the Automate Conference in Chicago, which allowed me to see the various forms of robotics that are used in industrial and manufacturing settings. This really taught me a lot about the way that robots are being incorporated into job environments that were previously occupied solely by humans. This also taught me about the concept of a “cobot” versus a “robot”. The difference between these two is that a “cobot” has been designed to cohabitate and work collaboratively with a human being, by incorporating various design features such as deal encoding for force sensing, ease of programming, and various machine vision algorithms.