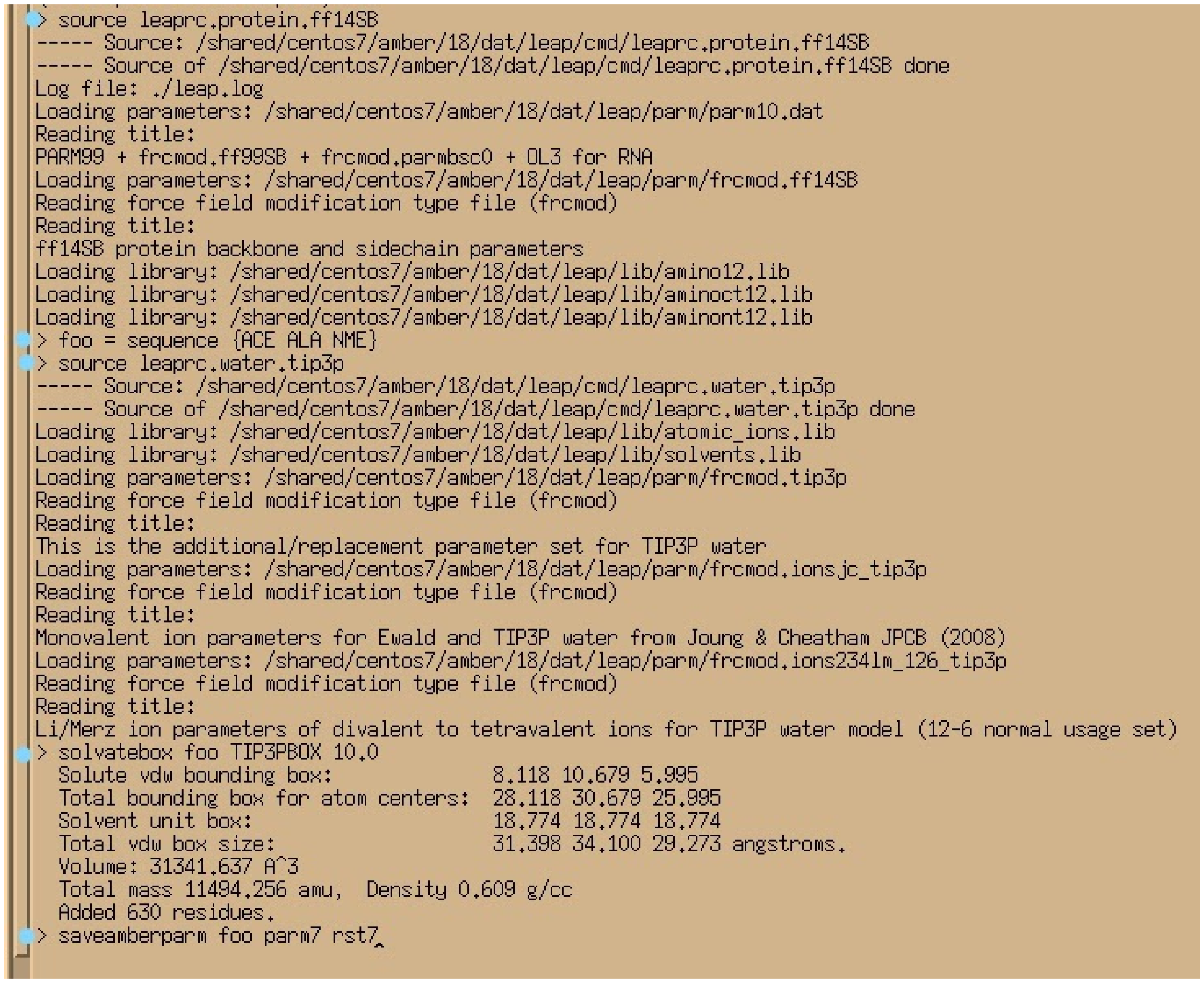
Pulmonary surfactant(PS) is a mixture of lipids and proteins that is essential to human life and proper lung function. It primarily functions in reducing the surface tension at the air-water interface in the alveoli, which prevents alveolar collapse as the lungs expand and contract. Additionally, pulmonary surfactant assists in defending against pathogens that enter through the lungs. Deficiency of this crucial substance, which is common in premature infants, leads to a condition called respiratory distress syndrome, which is fatal if not treated. Pulmonary surfactant is composed mostly of lipids, which form a monolayer at the air-water interface. Although these lipids make up the bulk of surfactant mass, the less abundant proteins are crucial in proper surfactant function. There are four distinct proteins that make up pulmonary surfactant, named surfactant proteins A-D. Surfactant proteins A and D are thought to function primarily in immune response, while proteins B and C allow rapid formation and reconstruction of the lipid monolayer as the lungs expand and contract. It is known that these proteins are crucial to these functions. Mice without these proteins show significantly reduced lung function. Surfactant treatments composed of only lipids are ineffective at treating respiratory distress syndrome. However, the structure and mechanism of action for these proteins is still largely unknown. Uncovering how these proteins function is an active area of research, and is an important step in understanding how the lungs react to damage and disease, as well as how we can develop effective treatments.

Surfactant Protein B(SP-B) is a relatively small 79-residue protein that is predominantly hydrophobic. It is a member of the saposin-like family of proteins, which are characterized by 3 internal disulfide bonds linking 4-5 helical regions. In pulmonary surfactant, SP-B is thought to function in the binding, lysis, and fusion of the lipid monolayer. Due to its small size and hydrophobic properties, SP-B is difficult to isolate and study. As such, there is currently no experimentally derived structure for this protein. So how do scientists study this molecule's structure and function?

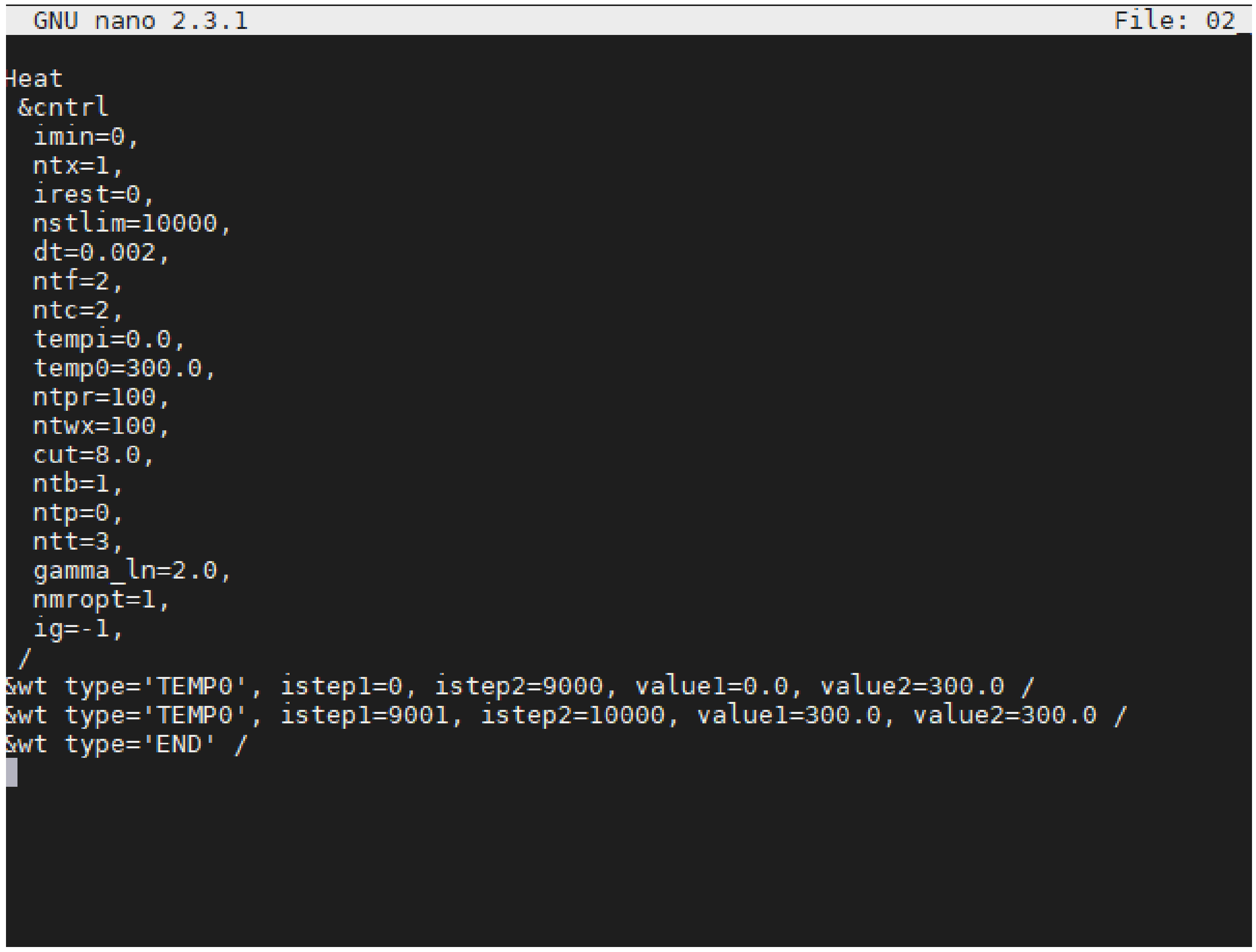
Molecular dynamics uses Newtonian mechanics along with numerical methods to simulate the physical changes of molecular systems over time. These simulations are extremely computationally intense, and are generally too much for a personal computer. For these simulations, I used the Northeastern Research Computing Discovery Cluster. Additionally, the numerical methods and algorithms used for determining forces and trajectories are numerous and well beyond the scope of this project. I used Amber 18, a suite of biomolecular simulation programs, to run the simulations and produce output data for analysis.

The first step of any MD simulation is to produce and load and prepare the model that is to be analyzed. Generally this model comes from the molecule's experimental structure. For molecules without an experimentally determined structure, like SP-B, alternative methods must be used. Structural models of SP-B can be created by extending upon a known functional fragment of SP-B(known as mini-B) using the secondary structure of a related molecule as a reference. This is called homology modelling. Due to the sudden shift to online instruction and the major interruption in research progress in March, I was not able to complete this step. From this point forward, I will be using a much simpler molecule, alanine dipeptide, to illustrate the process and develop some analysis scripts that should still apply when I am able to prepare a SP-B structure for simulation. The following 5 commands within Amber's Leap program load the protein force field that will be used, build the molecule that is going to be simulated, load the water model that will be used, solvate the molecule(add water molecules), and save this information to files that can be passed to the simulation program.



```
> source leaprc.protein.ff14SB
----- Source: /shared/centos7/amber/18/dat/leap/cmd/leaprc.protein.ff14SB
----- Source of /shared/centos7/amber/18/dat/leap/cmd/leaprc.protein.ff14SB done
Log file: ./leap.log
Loading parameters: /shared/centos7/amber/18/dat/leap/parm/parm10.dat
Reading title:
PARM99 + frcmod.ff99SB + frcmod.parmbsc0 + OL3 for RNA
Loading parameters: /shared/centos7/amber/18/dat/leap/parm/frcmod.ff14SB
Reading force field modification type file (frcmod)
Reading title:
ff14SB protein backbone and sidechain parameters
Loading library: /shared/centos7/amber/18/dat/leap/lib/amino12.lib
Loading library: /shared/centos7/amber/18/dat/leap/lib/aminoct12.lib
Loading library: /shared/centos7/amber/18/dat/leap/lib/aminont12.lib
> foo = sequence {ACE ALA NME}
> source leaprc.water.tip3p
----- Source: /shared/centos7/amber/18/dat/leap/cmd/leaprc.water.tip3p
----- Source of /shared/centos7/amber/18/dat/leap/cmd/leaprc.water.tip3p done
Loading library: /shared/centos7/amber/18/dat/leap/lib/atomic_ions.lib
Loading library: /shared/centos7/amber/18/dat/leap/lib/solvents.lib
Loading parameters: /shared/centos7/amber/18/dat/leap/parm/frcmod.tip3p
Reading force field modification type file (frcmod)
Reading title:
This is the additional/replacement parameter set for TIP3P water
Loading parameters: /shared/centos7/amber/18/dat/leap/parm/frcmod.ionsjc_tip3p
Reading force field modification type file (frcmod)
Reading title:
Monovalent ion parameters for Ewald and TIP3P water from Joung & Cheatham JPCB (2008)
Loading parameters: /shared/centos7/amber/18/dat/leap/parm/frcmod.ions234lm_126_tip3p
Reading force field modification type file (frcmod)
Reading title:
Li/Merz ion parameters of divalent to tetravalent ions for TIP3P water model (12-6 normal usage set)
> solvatebox foo TIP3PBOX 10.0
  Solute vdw bounding box:              8.118 10.679 5.995
  Total bounding box for atom centers:  28.118 30.679 25.995
  Solvent unit box:                     18.774 18.774 18.774
  Total vdw box size:                   31.398 34.100 29.273 angstroms.
  Volume: 31341.637 A^3
  Total mass 11494.256 amu,  Density 0.609 g/cc
  Added 630 residues.
> saveamberparm foo parm7 rst7
```

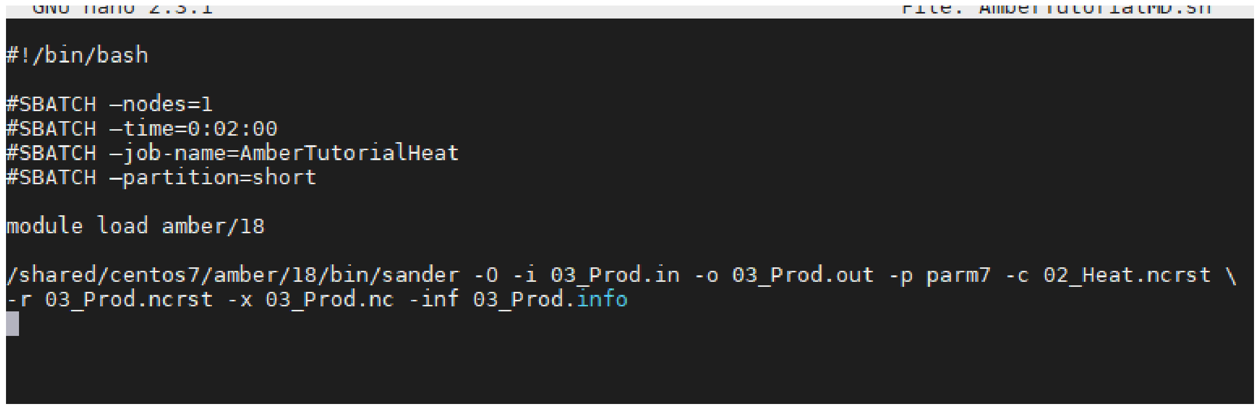
The next step was to prepare input files defining parameters for the simulation to follow. These include things like number steps, time between steps, cut-off distances for interactions, and controls on temperature, pressure or volume. The simulation is divided into 3 parts each of which take their own input files. They are energy minimization, heating from 0 to 300 k, and a production MD run at 300 k, and 1 atm. Below is an example of the heating input file.



```
  GNU nano 2.3.1                                                      File: 02

Heat
 &cntrl
  imin=0,
  ntx=1,
  irest=0,
  nstlim=10000,
  dt=0.002,
  ntf=2,
  ntc=2,
  tempi=0.0,
  temp0=300.0,
  ntpr=100,
  ntwx=100,
  cut=8.0,
  ntb=1,
  ntp=0,
  ntt=3,
  gamma_ln=2.0,
  nmropt=1,
  ig=-1,
 /
&wt type='TEMP0', istep1=0, istep2=9000, value1=0.0, value2=300.0 /
&wt type='TEMP0', istep1=9001, istep2=10000, value1=300.0, value2=300.0 /
&wt type='END' /
```

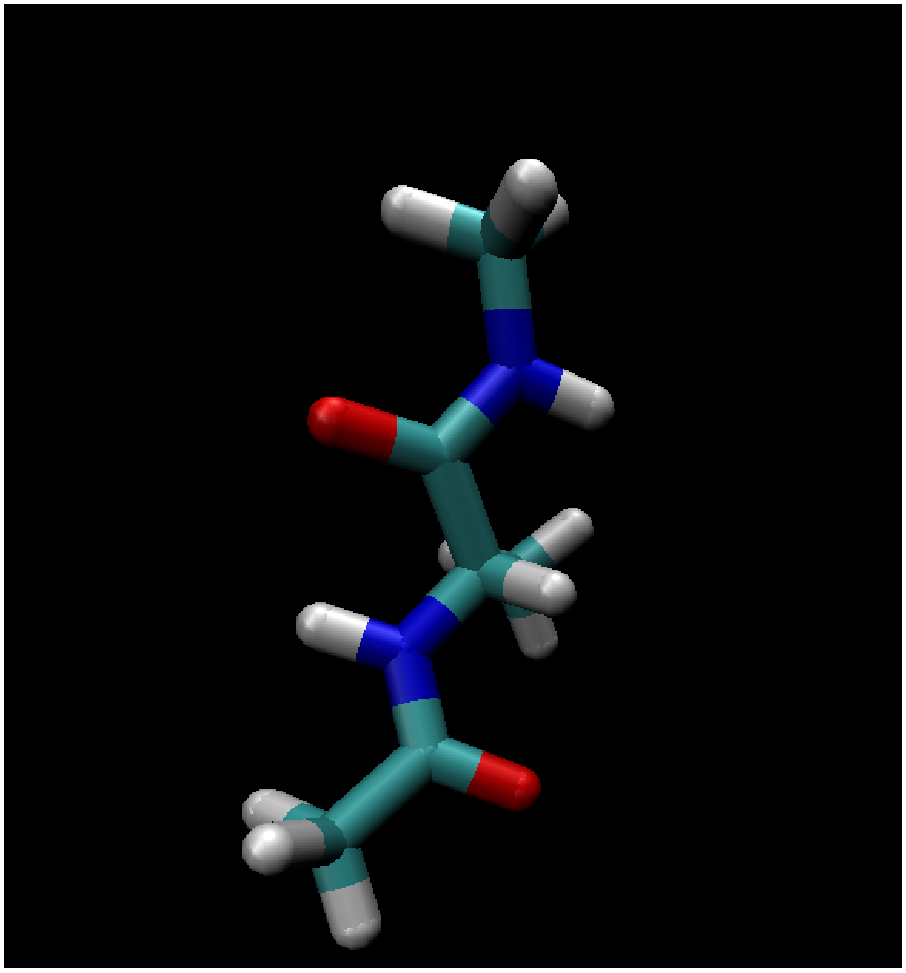
The parameters used here are taken from Amber Tutorial B0.

Next, we run these 3 simulations in order. These are run through the slurm job manager on the discovery cluster by using the command *sbatch script.sh*, where script is a short bash script that defines the job details(time, computing resources needed, etc.) and the commands that need to be run. Below is the script used for the production MD run. At the top, I specified the number and type of computing nodes, as well as the time needed and the job name. I then run the sander program, which executes the simulation, passing it the input files I created as well as the output from the heating run.



```
  GNU nano 2.3.1                                    File: AmberTutorialMD.sh

#!/bin/bash

#SBATCH —nodes=1
#SBATCH —time=0:02:00
#SBATCH —job-name=AmberTutorialHeat
#SBATCH —partition=short

module load amber/18

/shared/centos7/amber/18/bin/sander -O -i 03_Prod.in -o 03_Prod.out -p parm7 -c 02_Heat.ncrst \
-r 03_Prod.ncrst -x 03_Prod.nc -inf 03_Prod.info
```

These simulations output several files that contain large amounts of data on the simulation. I used the Amber script *process_mdout.perl* to extract these data points from the simulation in individual files which can be more easily used for analysis. I chose to use density, volume, temperature, and potential and kinetic energies. Additionally, these files can be passed to the VMD program, which generates nice visuals of the simulated system. Here is a model of Alanine Dipeptide


.

For my analysis I used Python, specifically the numpy, matplotlib, and pandas modules.

Explanation of code is in comments of .py file. I will collect all this into a jupyter notebook for submission. I kept it in docs for now for spell-check.

One thing I would like to add to this would be some way to quantify the way some of these variables converge. Since the molecule should be coming to some kind of equilibrium, the density and energy values should come to a roughly constant value. One thing I have seen is using the root mean square deviation of the data and seeing when it gets below a certain threshold value to determine when the values have converged.