Computational Biomolecular Simulation

1 Introduction

This pdf is supplement to lecture for biomolecular molecular dynamics simulations and should serve as a quick reference and guide you through an introduction into Molecular Dynamics (MD) simulation. It will utilize the AMBER (Assisted Model Building with Energy Refinement) simulation software. While their are many other software packages; like CHARMM (Chemistry at HARvard Macromolecular Mechanics), GROMACS (GROningen MAchine for Chemical Simulations), and NAMD (Nanoscale Molecular Dynamics) to name a few; that have different advantages and disadvantages, they all operate on the same underlying principles. Understanding how to approach a workflow in one will aid in the application of any of these programs and provide useful skills for interrogating molecular systems through a computational lens. By working through this series of tutorials you should

- Demonstrate competency and develop skills working within a Bash environment on Linux-based operating systems
- Prepare molecular dynamics simulation input, coordinate, and parameter/topology files for the AMBERMD software engine (sander)
- Model peptides, proteins, and nucleic acids in a molecular dynamics framework
- Visualize and analyze the results of MD simulations in VMD (Visual Molecular Dynamics)
- Calculate important observables from simulation outputs; such as total energy and RMSD (root-mean-square deviation)
- Critique the results of these calculations and reconcile differences between simulation trajectories to address common underlying issues in molecular dynamic simulation protocols

Questions or Concerns: erich.kuechler@msl.ubc.ca
Lab Emergancy: jennbui@gmail.com

2 Basics in Bash

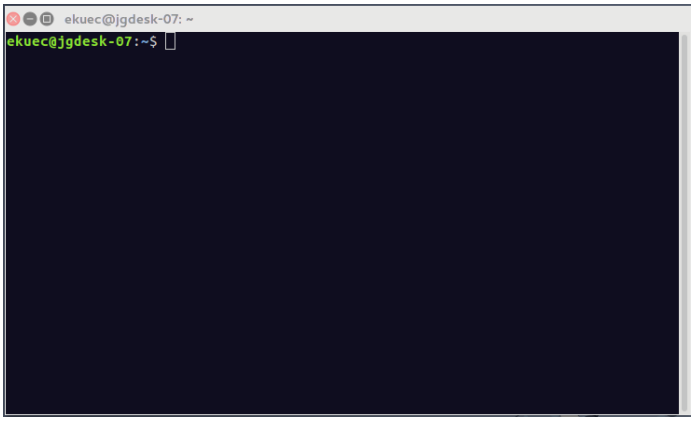
Most scientific software is made to operate on Linux-based systems. Linux provides an environment for scalable, secure, remotely-accessible computers and clusters with access to many open-source tools and programs to improve scientific workflow. As such, many (if not all) of the world's larges supercomputers utilize Linux to address the most challenging computational challenges to date. It is therefore important, not only to use the software required in the course but as a scientist, to gain a basic understanding on how to navigate within a Linux environment.

The first step, is to familiarize yourself with the most fundamental tools available to a computationalist. Unlike a majority of lay uses of computers, which function through GUIs (Graphical User Interfaces), much of scientific computation is performed through a *command-line interface*. Instead of using a mouse and clicking on folders and programs to perform functions, you will need to type commands into a *terminal* with the *Bash* programming language. There are other languages that are common, but Bash is the most widely used on Linux platforms and becoming familiar with its usage will give you transferable skills to other programming languages if you need to use them in the future. Command-line interfaces, or shells, have many advantages over GUIs in scientific computation. For scientific computation, the leading advantages include having traceable (recordable) precision in your work that can be scripted to perform many tasks far more quickly than possible with GUIs and being able to remotely access your work quickly and securely.

2.1 Opening Terminal

To begin, you will need to find the terminal icon on your system. It should be found on the far left hand side of your screen if you hover the mouse near the edge. This location is not universal, but is the default on the Ubuntu operating systems you will be using in this tutorial. The icon should look like

Click it to open a terminal window. Alternatively, you can use the `Ctrl+Alt+T` keyboard shortcut. A window should open which looks like



This shell will allow you to navigate, run programs, and edit files through various typed commands. For our purposes, we will be using the Bash programming language (the default in Ubuntu) and will cover several of the most useful in this tutorial. However, if you have questions, feel free to ask. Alternatively, there are many resources online that can serve as quick-start guides as well.

2.2 Moving around in Terminal

Upon opening the terminal you should see some text followed by your text cursor. In the figure above the text"`ekuec@jgdesk-07:~$`" is called the *command prompt*. For brevity, and the fact that this prompt may change depending on where you are in your file tree, we will signify it by "`[~]$`". When you are instructed to type commands, the information you will be typing will appear *after* this indicator. For instance,

```
[~]$ ls
```

means that you should literally type 'ls' and hit `Enter`. This command will then list the files and directories in your current location. Furthermore, many commands have additional options that can be used for greater utility. Such as

```
[~]$ ls -a
```

will list "all" in the current location, including hidden files. If you want to learn about options for built-in functions, you can access the manual pages individually with the `man` command. Such as

```
[~]$ man ls
```

will detail all of the options for `ls`. You can exit the manual page by typing 'q' on the keyboard. Options for commands can be stacked upon them seves

to give you the specific information you would like, for instance

```
[~]$ ls -al
```

will list 'all files in long form'.

For now, let's make a new *directory* by typing

```
[~]$ mkdir Tutorials
```

with this command you should **m**a**k**e a **dir**ectory named "Tutorials". Let's make two more directories

```
[~]$ mkdir junk1 junk2
```

note how we can make two directories at once within a single command. But those directories are junk, clearly, so let's now get rid of them with the command

```
[~]$ rmdir junk1 junk2
```

to **r**e**m**ove them. The `rmdir` command should only remove the directories specified and also only operate on empty directories. Otherwise, you *should* get a warning message.

Next, let's navigate into the remaining `Tutorials` directory. You can do this by typing

```
[~]$ cd Tutorials
```

which will cause you to **c**hange **d**irectory into Tutorials. Quickly, let's make another directory called "T0_Bash" then immediately move into it with

```
[~]$ mkdir T0_Bash; cd T0_Bash
```

As you see, we can issue multiple commands by separating them by a semicolon ";". The shell will interpret these commands in order from start to end. You should be a little careful if you issue multiple commands, however, because the shell will try to complete the command to the best of it's ability. Meaning that if a command early in the command string is erroneous then the remaining commands will still attempt to complete which may lead to problematic situations. Now, let's see where we are by typing

```
[~]$ pwd
```

The `pwd` command displays the **p**ath **w**orking **d**irectory. This command is particularly useful when moving around directory trees or moving files, which we will discuss in a bit. However, before we get to files, we still have to learn a couple things about moving around. Let's say we want to move back one

directory, out of the `T0_Bash` folder into the `Tutorials` directory. You could issue the command `cd ../` where '../' means to 'go up' one rung in the file tree. Instead, lets go two directories back with

```
[~]$ cd ../../
```

You should now be where we began. You can check your location with the `pwd` as well. To go back to where we were we can type the command

```
[~]$ cd Tutorials/T0_Bash/
```

note that you can start to type the directory names and auto-complete them by using the `Tab` key, which will auto-fill names of directories or files (depending on the command) as far as it can. Additionally, if you type

```
[~]$ cd
```

you will move to your home directory. If, at any point, you want to move into the directory in which were immediately previously in you were then type

```
[~]$ cd -
```

which can be useful if you are climbing long distances though files of accidentally `cd` to home when not meaning to.

2.3 Using Text Editors (Vim)

You should be in the Tutorials/T0_Bash/ directory now, if you are not please move there. Now, let's learn how to make new files and edit them. There are a variety of different text editors that you could use like the user friendly `gedit`, however, if you are planning on trying to use a compute cluster or to work remotely, then you will likely want to use Vim. DISCLAIMER: Vim is not easy to use initially, and while I will instruct you to use it throughout the tutorials, you can always use other editors (gedit should also be installed and work) that will be much more familiar to get started, but you Vim is commonly used for creating scripts and has many features which make creating and editing shell scripts easier. However, Vim is not the most user friendly text editor and some care should be taken when first getting used to the program. Specifically, Vim has three main "modes" of functioning: command mode, input mode, and last line mode. Command mode allows you to access built-in functions, some of which will be used in this introduction. Input mode allows the user to type and paste text. This will be the mode that you will use the most and works much like text editors you are likely familiar using. In last line mode you can utilize some of the most powerful commands, like regular expressions to find and replace text or to manipulate large amounts of data, however these functions are mostly for advanced users and only the very basics will be discussed here.

To create a file, type

```
[~]$ vi myfile.txt
```

which should open Vim. Vim will try to access the file "myfile.txt" however, if it does not exist it will assume that you are trying to create a new file with that name. To start typing, press `i`.This should put you into insert mode, and you should see `-- INSERT --` at the bottom of the screen. You should be able to type freely, so type something. Now you want to save this file. First you want to exit insert mode. Hit the `Esc` key, and the `-- INSERT --` at the bottom of the screen should vanish. You are now in command mode (the default mode when you open Vim). To save and quit you can type `ZZ` (`shift+z shift+z`) or you can do so by typing `:wq`. If you examine the files you have in your directory with the `ls` command, you should see that you have a new file in your directory that you just created. Let's open this file again, with the same command as before. You can either type out the fill command or hit the "up" arrow on your keyboard to scroll back through your history. The terminal will retain a history of the session for several commands, which can be very useful if you are troubleshooting.

Once the file is open type `$`. This command will bring your cursor to the lend of the line. Then hit `a`, which will move the cursor one more space to the right and activate insert mode (to append to the current line). Type a few more words then hit `Esc`. Now use the mouse to highlight the text you just wrote and then tap `o` which will move your cursor to the next line and active insert mode. Now click in the mouse wheel. This should copy the highlighted text onto the new line. In Linux, highlighted text is copied to the clipboard and clicking the mouse wheel (or on some systems a right click) will paste. Save and quit the document.

It should be noted that while Vim is a powerful text editor, it is not the most user friendly for beginners. If you cannot get Vim to work for you, then you might want to use the program gEdit by using the command

```
[~]$ gedit myfile.txt
```

which will open a more familiar GUI-based text editor, much like notepad. However, you will not be able to use GUI-based programs when you are accessing computers remotely (which you may want to do later to check your simulation progress later outside of the office).

2.4 Copying, moving, and removing files

If you want to copy a document you can through using the `cp` command

```
[~]$ cp myfile.txt mynewfile.txt
```

should create a new file that is exactly the same as the file you previously edited.

Keep in mind that you do not always need to specify a name for the new file, just a destination. If you want to copy a file one directory up with the same name, you can type "`cp myfile.txt ../`" or if you want to copy a file from another directory here then you can type "`cp /path/to/file/myfile.txt .`" where the "." means 'here'. Additionally, if you want to look at all of the files of a specific type you can use the command

```
[~]$ ls *.txt
```

which will `ls` all files with the `.txt` extension. Here the '*' character is called a *wildcard* and is very useful for managing large amounts of information. It will match anything, any number of times for the command to act upon. As such, *extreme* caution should be used when wildcard in conjunction with commands that alter files or directories; especially the next command we are going to learn.

You can delete a file by using the `rm` command

```
[~]$ rm mynewfile.txt
```

however be *very careful* when removing files. Unlike Windows, when you delete a file in Linux it will be gone. You account should have been altered to prompt you to make sure you want to delete the file. To confirm, enter "y". This behaviour is not default within Linux, the `rm` command has been replaced with `rm -i`, but it is generally good practice.

Very similar to the `cp` command is the `mv` command, which will **mo**v**e** a file. What this command accomplishes is effectively a copy followed by removing the original source file. Additionally, whenever you `cp` or `mv` a file, you will need to be careful to not overwrite files unintentionally. Bash will not warn you if the file you are writing to exists but rather will simply overwrite it if the name is the same.

2.5 Remote Access

If you would like to learn how to remotely access your workstation from your personal computer, please come and speak with me. While this process is not too complicated it does require some personalized instructions depending on your operating system and which workstation you are working upon. While remote access is useful, it should not be required for completion of the materials of the coarse.

2.6 Reporting Findings

Throughout the tutorials you will be expected to obtain some deliverables which will be used to assess your understanding and completion of the assignments. These should be PDFs that you will email to me (E. Kuechler) in which you will add images of plots made in xmgrace, added to libreoffice (a word processor on

your workstation) documents, and then commented upon. Details will either be included in the tutorials themselves or be forthcoming.

2.7 Linux Cheat Sheet

Command	Use	Comment
cd	cd ../dir/	Change Directory
ls	ls *.txt	list either all or all of a type
vi	vi file.txt	open the Vim text editor
mkdir	mkdir dir	make a directory (dir)
pwd	pwd	display working directory
cp	cp f1 f2	copy file1 to file2
mv	mv f1 f2	move file1 to file2
rm	rm f1	delete file1
ssh	ssh usr@host	connect to a remote host
scp	scp f1 usr@host:	copy to a remote host
locate	locate f1	find file1
grep	grep "foo" *.txt	search (all .txt) for foo

subsectionVim Cheat Sheet

Command	Use
i	Enter –INSERT– mode, which you can use to type
a	Enter –INSERT– mode, move cursor one place to the right
o	Enter –INSERT– mode, move the cursor to the next line
Esc	Exit –INSERT–
$	Move cursor to the end of the line
gg	Move the cursor to the first line
G	Move the cursor to the last line
? foo	Search for "foo", hit "n" for next result
dd	delete current line
d$	delete from cursor to the end of the line
dG	delete from cursor to the end of the document
ZZ	save and quit
:wq	save and quit
:w	save
:q!	force quit, if you really want to exit

3 Running a Molecular Dynamics Simulation

This tutorial is designed to provide an introduction to molecular dynamics simulations with Amber. It is designed for new users who want to learn about how to run Molecular Dynamics simulations. AMBER stands for Assisted Model Building and Energy Refinement. It refers not only to the molecular dynamics programs, but also a set of force fields that describe the potential energy function and parameters of the interactions of biomolecules. In order to run a Molecular Dynamics simulation in Amber, each molecule's interactions are described by a molecular force field.

To begin, you will want to move into your `Tutorials` directory and make yourself a new folder for your own calculations. One challenge for scientific computation is having an easy to understand system of well organized files; a mundane challenge but one that becomes extremely important as you begin to deal with thousands of different files, many of which may be very similar while only altering one or two conditions. The previous tutorial *implied* a naming convention that you can use, however use one that makes sense to you and that you can keep track of. Additionally, most of the commands listed in this tutorial will assume a specific naming convention for the individual files, however you are free to change them as you see fit as long as you are able to follow along.

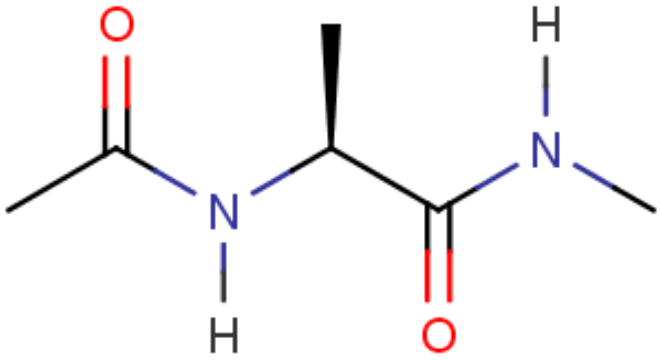
In order to run an MD simulation with sander, the molecular dynamics engine in AMBER, three key files are needed:

1. prm7 - The file that describes the parameter and topology of the molecules in the system
2. rst7 - The file that describes the initial molecular coordinates of the system
3. mdin - The file that describes the settings for the Amber MD engine

The prm7 file contains a subset of parameters in the sourced AMBER *force field* and how the atoms in the system are connected to each other (the topology). The parameters in the force field are related to the bonds, angles, torsions, electrostatics, and Lennard-Jones terms; all of which together describe a Hamiltonian (a potential energy function). Molecular dynamics works by allowing system to sample within the Hamiltonian in order to gain information. The rst7 file provides the initial coordinates of all of the atoms in your simulation. Given these coordinates (and sometimes initial velocities) and the Hamiltonian, you are providing the molecular dynamics engine the initial conditions and the energy landscape to propagate upon. The last file, mdin file, then tells sander (the molecular dynamics engine) how to run and which settings to use.

3.1 Prepare topology and coordinate files

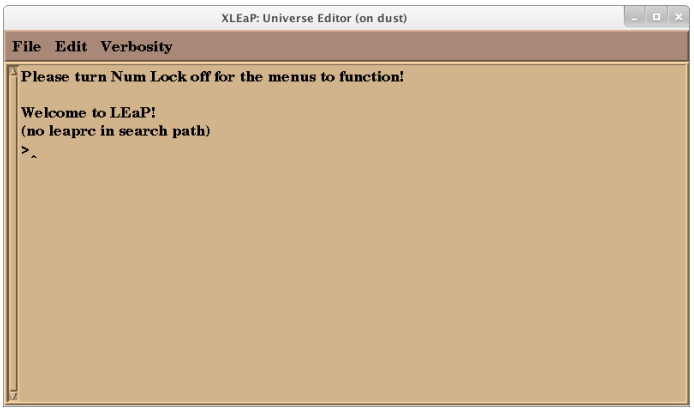
For our first simulation we are going to use a classic system that is frequently used as a prototypical example in molecular dynamics: alanine dipeptide.



To build and solvate this molecule we will use a built-in AMBER program known as xLEaP. xLEaP has a command line interface, much like the Bash terminal, while also having a simple molecular graphics interface for building topology and parameters files for molecules. While it is similar to the Bash terminal, the commands will be very different and the terminal-like GUI is far less forgiving than a normal shell. You cannot press the arrow keys to scroll through your history or move the text cursor back along your text to correct typing mistakes. Start xLEaP now with the `xleap` command

```
[~]$ xleap
```

You should see a window like



Warning: Do NOT click the "X" on any LEaP window. It will quit LEaP entirely. Also, some versions of xLEaP require you to turn Num Lock off for the menus to work. With xLEaP open we can now start to create our system that

we will simulate later. Commands within the xLEaP GUI will be designated with the '¿' symbol.

3.2 Load a protein and nucleic acid force field

A MD force field relates the parameters that describe the intra- and intermolecular interactions between the molecules in the system. In MD, this Hamiltonian (potential energy function) is integrated to describe the forces and velocities of the molecules. In order to run a molecular dynamics simulation, we need to load a force field to describe the potential energy of alanine dipeptide. We will use the AMBER force field FF14SB for proteins and nucleic acids. We need to source (load) the FF14SB force field, so within xLEaP type

```
> source leaprc.protein.ff14SB
```

note that LEaP "knows" where the file is, despite not explicitly being told. An *environmental variable* was set and used when you loaded the program so that the program has access to databases that it can use. If however, you wanted to load a different force field you could specify a full path to a personally modified file.

3.3 Build alanine dipeptide

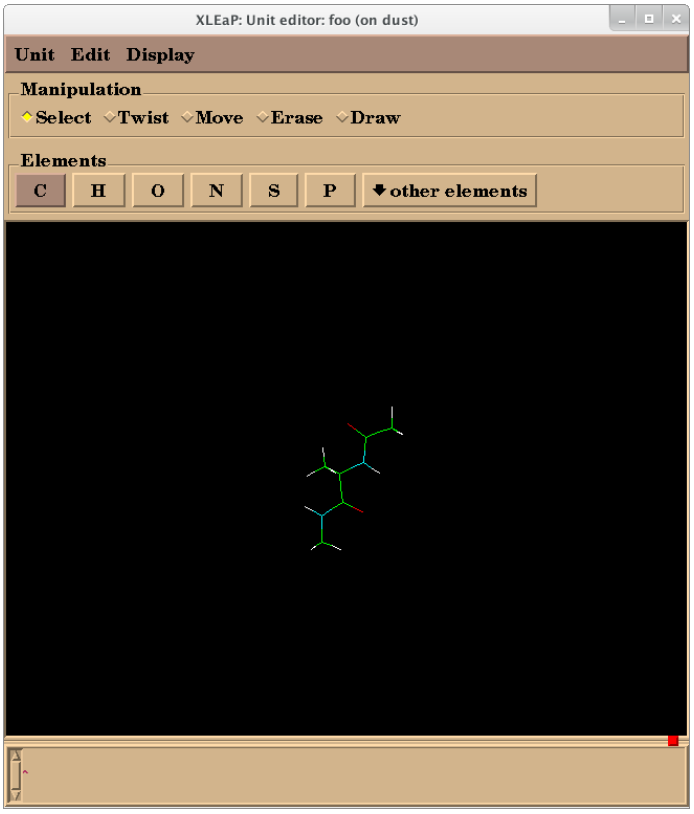
We can build an alanine dipeptide as an alanine amino acid capped with an acetyl group on the N-terminus and n-methylamide on the C-terminus. After we loaded the force field ff14SB, xLEaP now has these "units" available to build into a molecule. The sequence command will create a new unit from the available units and connect them together. Use `sequence` to create a new unit called `myMOL` out of the `ACE`, `ALA`, and `NME` units. Within xLEaP

```
> myMOL = sequence { ACE ALA NME }
```

Now you have a single alanine dipeptide molecule stored in the unit `myMOL`. xLEaP provides a very basic editor to examine and change units and molecules. We can examine the structure of the alanine dipeptide molecule. Use the `edit` command to view the structure

```
> edit myMOL
```

The editing window will look like this:



From here, you can examine the topology, structure, atom names, atom types, and partial charges of the molecule. Basic editing of the molecule is also possible. Warning: Do NOT click the "X" to close this window. It will exit xLEaP. To close this window, use Unit → Close.

3.4 Solvate alanine dipeptide

The next step to prepare this alanine dipeptide system is to solvate the molecule with explicit water molecules. In this simulation we will use add TIP3P water molecules to the system. We will place the system into a solvation box with periodic boundary conditions, meaning that molecules that exit one side of the system will wrap to the other side of the system. It is important that the periodic box is large, i.e. there is enough water surrounding alanine dipeptide, so that the alanine dipeptide molecule does not interact with its periodic images, but making the periodic box too large will make the simulation take a very long time and it would become computationally expensive. There are many different water models available for MD simulations. However, for this tutorial we will use the TIP3P water model.

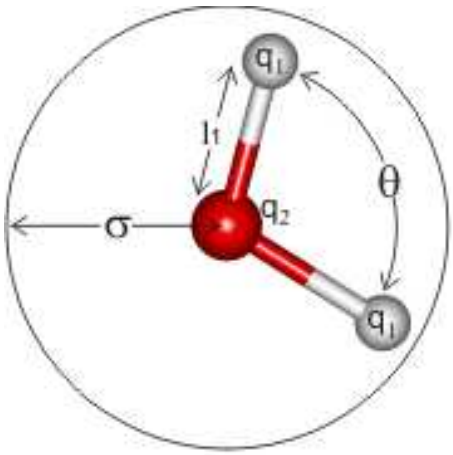


Figure 1: Modified from http:www1.lsbu.ac.ukwaterwater_models.html

TIP3P water is a rigid water model where the charges of hydrogens and the oxygen are fixed at their atomic centers and the volume of the entire molecule is described as a sphere centered at the oxygen. We can solvate the system with the `solvatebox` command after sourcing the TIP3P force field parameters.

```
> source leaprc.water.tip3p
```

then

```
> solvatebox myMOL TIP3PBOX 10.0
```

TIP3PBOX specifies the type of water box to solvate with. 10.0 indicates that the molecule should have a buffer of at least 10 Angstroms between alanine dipeptide and the periodic box wall. These solvent boxes are pre-equilibrated units within AMBER that have an artificially low density. Waters are removed from the box to 'fit' your solute into the center of the box while the low density will allow the water to relax around your molecule of interest during minimization and equilibration.

3.5 Save the Amber prm7 and rst7 input files

Now we will save the myMOL.prm7 and myMOL.rst7 files to the current working directory. The unit myMOL now contains the alanine dipeptide molecule, water molecules, and the periodic box information necessary for simulation. The parameters will be assigned from the ff99SB force field. To save the prm7 and rst7 file use the `saveamberparm` command. The command works by

```
saveamberparm <UNIT> <PRM7> <RST7>
```

so in order to save our files we will want to type

```
> saveamberparm myMOL myMOL.prm7 myMOL.rst7
```

and should produce output

```
Checking Unit.
Building topology.
Building atom parameters.
Building bond parameters.
Building angle parameters.
Building proper torsion parameters.
Building improper torsion parameters.
total 4 improper torsions applied
Building H-Bond parameters.
Incorporating Non-Bonded adjustments.
Not Marking per-residue atom chain types.
Marking per-residue atom chain types.
Residues lacking connect0/connect1 -
these don't have chain types marked:

    res total affected

    WAT 630

(no restraints)
```

Warning: Pay close attention to the output of this command for any warnings or errors that your prm7 and rst7 file did not build correctly. If errors happen, try to repeat the process and if that does not work then please ask for assistance.

Once the files are successfully built, you can quit xLEaP with the command

```
> quit
```

3.6 Prepare Amber MD sander input files

The last components we need are the input files that define the program settings for each MD run. For this system, we will perform an energy minimization on the system, then slowly heat the system, and then do production MD at the desired temperature and pressure.

1. Minimization

2. Heating with constant volume and temperature (NVT) for 20ps from 0K to 300K

3. Production MD with constant pressure and temperature (NPT) at 300K and 1atm for 60ps

We will save the trajectory and write to the output file every 2ps. The Langevin thermostat will be used to control the temperature. The random number generator will be initialized with a random seed. We first must create our input files.

3.7 Minimization Input

The first thing we will need to do with our new coordinates will be to minimize the system. While you could potentially skip this step and move directly into equilibration, if you have any *bad contacts* (such as overlapping atoms) then a equilibration under molecular dynamics will 'kick' those contacts very hard and lead to erroneous behaviour. Minimization will more gently remove any bad contacts and will tend towards a more well-behaved simulation. One important note, however, is that if you minimize your system for 'too long' (what is too long? who knows?) then you will introduce more bad contacts! Minimization, in AMBER, does not use periodic boundary conditions (mirror images of the system to represent bulk solution) and waters could accidentally drift into locations that their reflections would occupy once the periodic conditions are turned on. To create a minimization input file, use Vim or gEdit to create a file "01_Min.in" that includes:

```
Minimize
&cntrl
imin=1,
ntx=1,
irest=0,
maxcyc=2000,
ncyc=1000,
ntpr=100,
ntwx=0,
cut=8.0,
/
```

Where the input can be summarized as

imin=1	Choose a minimization run
ntx=1	Read coordinates but not velocities from ASCII formatted rst7 coordinate file
irest=0	Do not restart simulation. (not applicable to minimization)
maxcyc=2000	Maximum minimization cycles
ncyc=1000	The steepest descent algorithm for the first 0-ncyc cycles, then switches the conjugate gradient algorithm for ncyc-maxcyc cycles
ntpr=100	Print to the Amber mdout output file every ntpr cycles
ntwx=0	No Amber mdcrd trajectory file written (not applicable to minimization)
cut=8.0	Nonbonded cutoff distance in Angstroms

3.8 Heating input

After minimization, we will need to 'heat' our system. As the system currently stands, we have a block of ice (with the wrong density) at 0K with a peptide in it. As such, we need to add kinetic energy into our system. In this tutorial, we will use a Langevin thermostat to accomplish this task. In a nearly criminally short and practical explanation, we will randomly 'hit' our molecules with a small force in a random direction. To account for this increase in energy we will also uniformly bleed energy from the system by including a viscus damping term, so our molecules will effectively move around like they are in a thin syrup. By balancing the frequency and the strength of our artificial collisions with our damping we can control, within a certain amount of noise, the temperature of the system.

Additionally, when heating, we want to ramp up the temperature slowly. If we introduce energy too rapidly then we could move the structure of our solute into a non-native, higher energy state by jumping over normally insurmountable energy barriers. The system could then become trapped in this state and sample the incorrect ensemble of structures, leading us to erroneous conclusions. The rate at which you heat a system can depend on what you are simulating, but there are generally standard protocols which we can follow. To make an input for heating, create the file "02_Heat.in" that includes the following settings for heating:

```
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' /
```

Where the input can be summarized as

imin=0	Choose a molecular dynamics (MD) run
nstlim=10000	Number of MD steps in run (nstlim * dt = run length in ps)
dt=0.002	Time step in picoseconds (ps). The time length of each MD step
ntf=2	Setting to not calculate force for SHAKE constrained bonds
ntc=2	Enable SHAKE to constrain all bonds involving hydrogen
tempi=0.0	Initial thermostat temperature in K
temp0=300.0	Final thermostat temperature in K
ntwx=1000	Write Amber trajectory file mdcrd every ntwx steps
ntb=1	Periodic boundaries for constant volume
ntp=0	No pressure control
ntt=3	Temperature control with Langevin thermostat
gamma_ln=2.0	Langevin thermostat collision frequency
nmropt=1	NMR restraints and weight changes read (see NMROPT section)
ig=-1	Randomize the seed for the pseudo-random number generator

The final three lines allow the thermostat to change its target temperature throughout the simulation. For the first 9000 steps, the temperature will increase from 0K to 300K, effectively setting the temperature gradient. For steps 9001 to 10000, the temperature will remain at 300K, allowing the system to equilibrate at our desired temperature.

3.9 Production input

The production time of this simulation is only 60ps. Ideally, we would run this simulation for much longer, but in the interest of time for this tutorial, we have limited the production simulation time. Additionally, NTPR and NTWX are set very low so that it is possible to analyze this short simulation. Using these settings for longer MD simulations will create very large output and trajectory files and will be slower than regular MD. For real production MD, you'll need to increase NTPR and NTWX. As such, this input file is not intended for general MD simulations. We now want to create the file "03_Prod.in" with the settings for production MD

```
Production
&cntrl
imin=0,
ntx=5,
irest=1,
nstlim=30000,
dt=0.002,
ntf=2,
ntc=2,
temp0=300.0,
ntpr=100,
ntwx=100,
cut=8.0,
ntb=2,
ntp=1,
ntt=3,
gamma_ln=2.0,
ig=-1,
/
```

The settings for production can be summarized as follows:

ntx=5	Read coordinates and velocities from unformatted rst7 coordinate file
irest=1	Restart previous MD run
temp0=300.0	Thermostat temperature. Run at 300K
ntb=2	Use periodic boundary conditions with constant pressure
ntp=1	Use the Berendsen barostat for constant pressure simulation

3.10 Run Amber MD sander

Now that we have all the ingredients: the parameter and topology file prm7, the coordinate file rst7, and the input files 01_Min.in, 02_Heat.in, 03_Prod.in, we are ready to run the actual minimization, heating, and production MD. We will use the program sander, the general purpose MD engine of Amber. On the command line, we can specify several more options and choose which files are to be used for input. To fun the minimization we can use the command

```
[~]$ $AMBERHOME/bin/sander -O -i 01_Min.in -o 01_Min.out -p myMOL.prm7
-c myMOL.rst7 -r 01_Min.ncrst -inf 01_Min.mdinfo
```

where

-O	is a flag to overwrite the output files if they already exist
-i 01_Min.in	Indicates the input file (default mdin)
-o 01_Min.out	Write output file (default mdout)
-p myMOL.prm7	Choose parameter and topology file prm7
-c myMOL.rst7	Choose coordinate file rst7
-r 01_Min.ncrst	Write output restart file with coordinates and velocities (default restrt)
-inf 01_Min.mdinfo	Write MD info file with simulation status (default mdinfo)

sander should complete the minimization in a moderate amount of time (27 seconds) depending on your computer specifications. After sander completes, there should be an output file 01_Min.out, a restart file 01_Min.ncrst, and a MD info file 01_Min.mdinfo. You will use the restart file 01_Min.ncrst for the heating of the system. In the 01_Min.out file, you will find the details of your minimization. You should be able to see the system energy ENERGY decrease throughout the minimization. You can visualize this decrease in energy my creating a new file, *01_Min.ene*, with the following command

```
[~]$ grep -A1 "ENERGY" 01_Min.out | grep -v "ENERGY" | grep " " | awk '{print$1,$2}' > 01_M
```

and then you can plot this file with

```
[~]$ xmgrace 01_Min.ene
```

to see how the energy has changed over the minimization process.

After the minimization is complete we can use the restart file from the initial minimization to heat the system. In order to perform this simulation we have to run the command

```
[~]$ $AMBERHOME/bin/sander -O -i 02_Heat.in -o 02_Heat.out -p myMOL.prm7
-c 01_Min.ncrst -r 02_Heat.ncrst -x 02_Heat.nc -inf 02_Heat.mdinfo
```

where

-c 01_Min.ncrst	Now for the input coordinates we choose the restart file from minimization
-x 02_Heat.nc	Output trajectory file for MD simulation (default nc)

sander should complete the heating in a moderate amount of time (2.5 mins) depending on your computer specifications. Once it is finished, open 02_Heat.out in Vim. Some important values include

NSTEP	The time step that the MD simulation is at
TIME	The total time of the simulation (including restarts)
TEMP	System temperature
PRESS	System pressure
Etot	Total energy of the system
EKtot	Total kinetic energy of the system
EPtot	Total potential energy of the system

Note that the pressure is 0.0 because the barostat (pressure control) is not being used in the heating. Finally, now that minimization and heating are complete. We move on to the actual production MD. We need to use the command

```
[~]$ $AMBERHOME/bin/sander -O -i 03_Prod.in -o 03_Prod.out -p myMOL.prm7
-c 02_Heat.ncrst -r 03_Prod.ncrst -x 03_Prod.nc -inf 03_Prod.info
```

This final production run in the tutorial will take around 10 minutes.

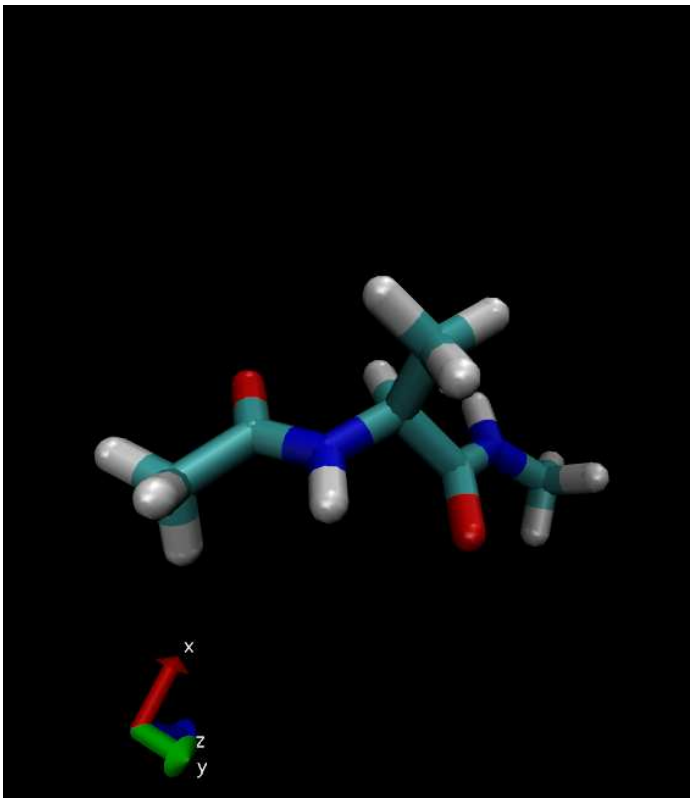
3.11 Visualize the results

You've now run an MD simulation. In order to visualize the results, we will now use a program called VMD (Visual Molecular Dynamics). This is a molecular graphics program that can render molecular structures in 3D. VMD not only loads Protein Database (PDB) structure files, but also MD trajectories from many programs. To start VMD, open a terminal, change directory to your tutorial files in the Tutorial directory, and run vmd.

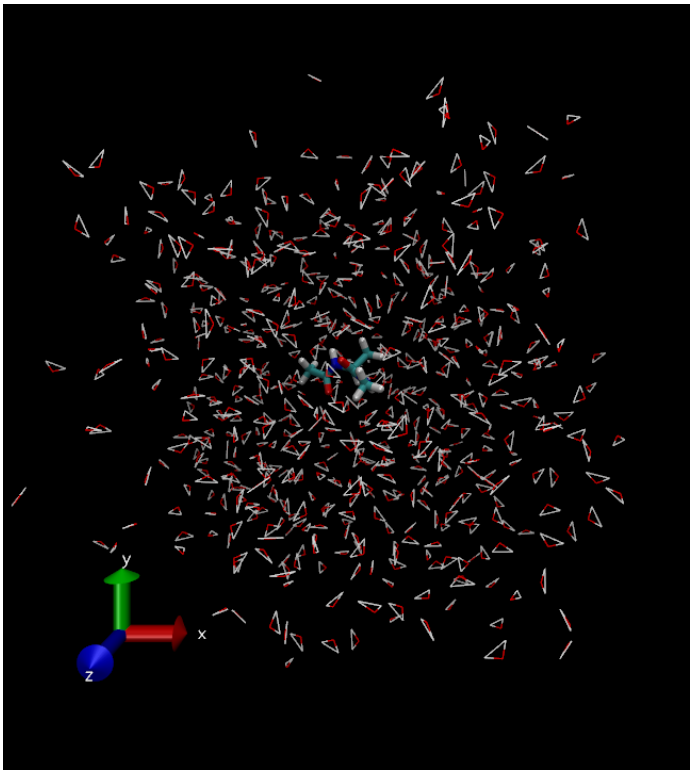
```
[~]$ vmd
```

Now we will load our MD trajectory to look at the dynamics of alanine dipeptide. Create a new molecule with File → New Molecule. Load files for New Molecule. Then choose the Amber parameter and topology file myMOL.prm7. Set the file type to AMBER7 Parm. Click Load. This will create an entry in the main VMD window called 0: myMOL.prm7. Click this entry to select it and then right click → Load Data into Molecule. Choose the Amber trajectory file 03_Prod.nc and set the file type to NetCDF (AMBER, MMTK). Click

Load. VMD now loads your trajectory to be visualized. The main VMD window can be used to control playback. You should be able to see the alanine dipeptide molecule as well as the many water molecules in the display. You can rotate, zoom and pan the molecules around the display with the mouse. Many different visualization options can be changed in the Graphics → Representations window. Your visualization can be restricted to the alanine dipeptide as well. Change the Selected Atoms to all not water. You can change the drawing method for the molecule to a more interesting model. Change the Drawing Method to Licorice. Alanine dipeptide will look something like this



Additionally, you can layer representations. Graphics → Representations and change the Selected Atoms to all. Now you can see the water molecules along with the stylized alanine dipeptide. Note that the water molecules have an artificial H-H bond. This bond is used in the AMBER force field to force the TIP3P water model to maintain geometry.



VMD has a lot of functions that can be used to analyze and study a MD trajectory. For example it is possible to align molecules, measure root mean squared deviations (RMSD), save structures from a trajectory, and measure physical system parameters throughout a trajectory. It's also possible to render a movie of a trajectory. However, these functions are beyond the scope of this introductory section. For more details refer to the VMD tutorial on the main AMBER Tutorial page.

3.12 Analyze the MD results

AMBER includes a suite of tools to examine and analyze MD trajectories. In this tutorial, we will do a simple analysis with several Amber programs and plot the results. The analysis will primarily be done from the command line in the terminal. Let's first make an Analysis directory and move into it

```
[~]$ mkdir Analysis; cd Analysis
```

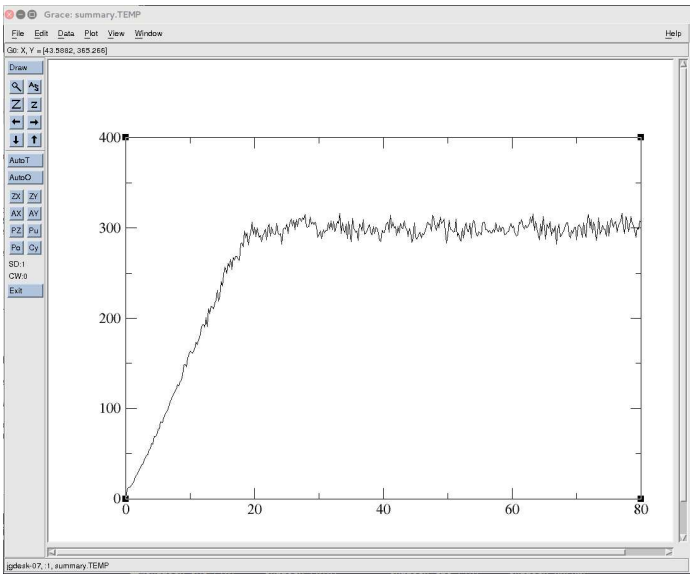
Then we can use some of the pre-made analysis scripts that are included with AMBER

```
[~]$ $AMBERHOME/bin/process_mdout.perl ../02_Heat.out ../03_Prod.out
```

It is now quite simple to plot the data saved in the output files. We will use a convenient, simple plotting program called xmgrace to automatically generate plots for the following MD simulation properties throughout the simulation. We use this for our convenience, but you can use any plotting program of your choice. We can look at the temperature during heating and production with the command

```
[~]$ xmgrace summary.TEMP
```

and you should get something that looks like



That shows how the temperature increases during heating and then remains fairly stable during the production simulation.

Deliverable: Now, attach your temperature plot as well as one containing the total energy (ETOT), kinetic energy(EKTOT), and potential energy(EPTOT) which you can make in xmgrace. In order to make a PNG or JPG with xmgrace you will have to use 'File→Print setup', select your format under the 'Device' tab, edit information that you like (dpi, output name, etc), then click 'Accept'. After which select 'File→Print'. Think critically and comment on the changes in these observables and why we are seeing the general trends. When you are finished, export the file as a PDF which has the filename format: SectionNumber_LastName_LastName_ADPOut.pdf (e.g. 01_Kuechler_ADPOut.pdf). Attach it in an email and send it to me.

Congratulations! You've run your first complete MD simulation and suc-

cessfully analyzed the results. This tutorial is a fairly simple example of the workflow for setting up, running, and analyzing your own MD simulation. You should now have the basic skills to begin working with simulations, however this workflow is highly simplified and would need to be further expanded and tested to produce publication-quality results.

If you got this far and want some fun, and thought provoking, extra steps to look at for your simulations here are two *optional* (or possibly optionally assigned) objectives that you can attempt:

- Perform another minimization, but this time increase the number of minimization steps to 4000, think about and comment on the differences between the decrease in energy between this new minimization and the original. Would it be practical to use this longer procedure? Was the original sufficient?

- Perform the MD again but at a much higher temperature, something completely unreasonable, like 800K. Frequently, computationalists use high-temperature simulations to 'enhance sampling'. However, care must be taken when doing so and we are not being careful here. Compare the old and new trajectories in VMD. A useful tool would be to use the tool in 'Extensions→Analysis→Ramachandran plot'. Don't forget to select your parameter file in the Molecule tab and watch the video.

4 Solvation and Electrostatics with Simulations of DNA

In this tutorial, we will build on our skills of the previous assignment by running another molecular dynamics simulation. However, to differentiate the two, we will examine the effects of long-range electrostatics and solvation models. Additionally, instead of a peptide system we will be examining a nucleic acid system which we will set up from a PDB (Protein Data Bank) file. Finally, we will perform some basic calculations common to MD trajectory analysis.

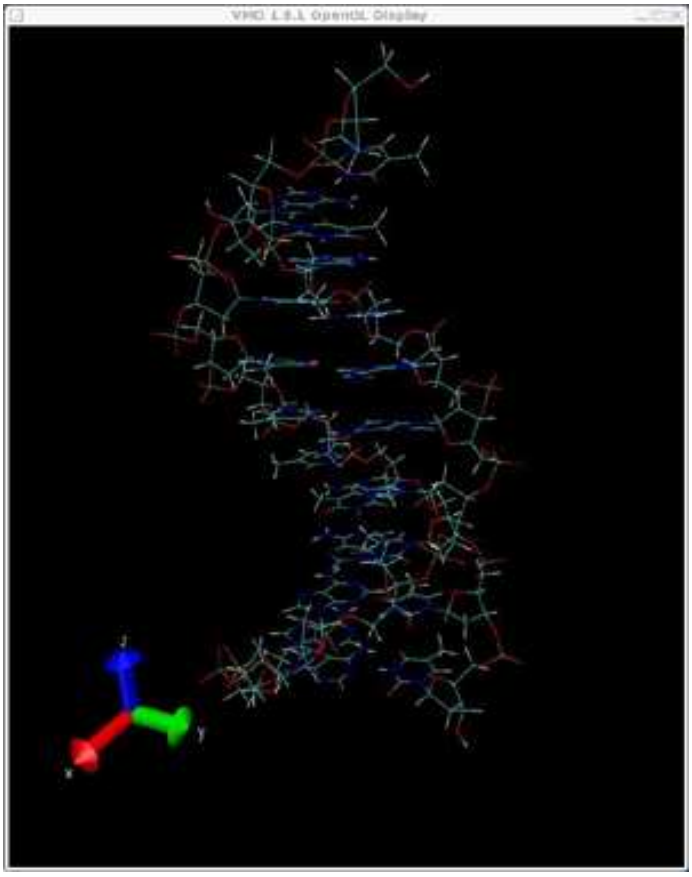
Therefore our objectives are

- Loading Protein Data Bank structures into LEaP to generate parm7 and rst7 files
- Minimize, heat, and run Molecular dynamics simulations
- Compare *in-vacuo* and implicit solvent simulations
- Visualize results in VMD
- Calculate root-mean-squared deviations and other energy functions over time

4.1 Setting up duplex DNA: polyA-polyT

Again, the first step in any modeling project is to develop the initial model structure. In principle, we could use xleap to build a model structure by hand as we did in the previous tutorial. However, when systems are very large or complex it is generally not practical to use this approach. Additionally, biologically relevant structural information could be hard to acquire when building systems from scratch. These issues can be alleviated by using experimentally determined structures obtained through databases of crystal or NMR structures such as the Protein Data Bank or the Cambridge Structural Database. With nucleic acids, users can also search the Nucleic Acid Database.

Unfortunately, we do not have a long time to run through these tutorials so we will not use a proper experimental structure. Instead a 10-mer DNA duplex PDB file has been made for you. In this way, you will know the fundamental ideas behind using PDB files to generate structures without needing to go through the tedious process of correcting bad or missing contacts from experimental data while also sifting though missing/ill-defined parameters.



Lucky you. A file should be located in your home directory under ref. To copy that file here you can simply type the command

```
[~]$ cp ~/ref/DNA.pdb .
```

It is always a good idea to look at the models before trying to use them. Problems can often be identified by quickly examining structures before trying to run expensive calculations only to have them crash or yield poor data. xLEaP can display models assuming that appropriate residue definition files are loaded and the residue names in the PDB file are consistent with those definitions.
We will build 3 different DNA models: an *in vacuo* model of the poly(A)-poly(T) structure (named polyAT_vac), an *in vacuo* model of the poly(A)-poly(T) structure with explicit counter-ions (named polyAT_cio), and a TIP3P explicitly solvated model of the poly(A)-poly(T) structure in a periodic box (named polyAT_wat). The *in vacuo* model will be applied in simulations to get a feel for MD and then the solvated model will be used for periodic boundary

simulations using a particle mesh Ewald treatment. The *in vacuo* model with the explicit ions will not be used for simulation but it is a good idea to build it in case it is needed for later analysis.
First we need to load xLEaP

```
[~]$ xleap -f leaprc.DNA.bsc1
```

where leaprc.DNA.bsc1 is a specific force field parametrization for DNA. Additionally, we must load the ion parameters for TIP3P water for what we will do later, using the command:

```
> source leaprc.water.tip3p
```

We can now load a PDB into xLEaP using the loadpdb command. This will create a new unit in xLEaP and load the specified PDB into that unit. We can then subsequently view and edit the new unit using the edit command. To load a PDB file into a new unit called "dna1" type the following in the xLEaP window (make sure the xleap window is highlighted and your mouse cursor is within the window):

```
> dna1 = loadpdb "DNA.pdb"
```

To look at the structure in xLEaP you can use the `edit` command. You will have to specify the unit you wish to edit; which is the one we just created

```
> edit dna1
```

This should open the xLEaP editor window and display the molecule. Within this window the LEFT mouse button allows atom selection (drag and drop), the center mouse button rotates the molecule and the RIGHT mouse button translates it. If the center button does not work you can simulate it by holding down the Ctrl key and the LEFT mouse button. To make the image larger or smaller, simultaneously hold down the center and RIGHT buttons (or Ctrl and RIGHT) and move the mouse up to zoom in or down to zoom out. If you played with selecting atoms using the left mouse button you can deselect a region by holding down the Shift key while drawing the selection rectangle. To select everything, double click the LEFT button (and to deselect, do the same while holding down the Shift key). Take a look at the structure. You should be able to see the perfect symmetry of canonical B-form geometry DNA. The perfect symmetry of canonical duplexes is based on analysis of long fibers of DNA. Real nucleic acids do not necessarily adopt this perfect symmetry as will become apparent once we start to carry out molecular dynamics on this 10-mer. WARNING: when you are finished looking at your structure, recall that you *should not* click the 'x' to close but rather should use the menu system or xLEaP will entirely close. To create the prm7 and rst7 files, we issue the commands we learned in the previous tutorial using the main xLEaP window:

```
> saveamberparm dna1 polyAT_vac.prm7 polyAT_vac.rst7
```

You will get the following output and you should ignore the warning. It concerns the fact that we did not neutralize our system.

```
Checking Unit.
WARNING: The unperturbed charge of the unit:  -18.000000 is not zero.

-- ignoring the warning.

Building topology.
Building atom parameters.
Building bond parameters.
Building angle parameters.
Building proper torsion parameters.
Building improper torsion parameters.
total 110 improper torsions applied
Building H-Bond parameters.
Not Marking per-residue atom chain types.
Marking per-residue atom chain types.
(no restraints)
```

Recall that prm7 files are parameter/topology files that define the connectivity and parameters for our current model. This information is fixed and does not change during the simulations that we will be running. The rst7 files are the atomistic coordinates and might optionally contain box coordinates and velocities. The information in this file sets initial conditions and will change during the simulation.

With are first system prm7 and rst7 files created, we now want to make a system that contains explicit net neutralizing counter-ions. There are a number of different ways to add ions to a structure, however, we will simply use the built-in `addions` command implemented in xLEaP. This method works by constructing a Coulombic potential on a 1.0 angstrom grid and then placing counter-ions one at a time at the points of lowest/highest electrostatic potential.

```
> addions dna1 Na+ 0
```

Will add sodium cations using this protocol until '0' net charge is reached. This procedure should therefore add a total of 18 sodium ions to counteract the -18 charge of the DNA chain. We are again ready to write the prm7 and rst7 files, this time for our neutralized system

```
> saveamberparm dna1 polyAT_cio.prm7 polyAT_cio.rst7
```

Lastly, we want a solvated DNA with explicit counter-ions. We have our "dna1" unit already built with counter-ions so the next step is to solvate it with explicit

water. Recall that we can solvate a system with the command `solvatebox`. For our DNA, we will put an 8 angstrom buffer of TIP3P water around the DNA in each direction.

```
> solvatebox dna1 TIP3PBOX 8.0
```

and then save our prm7 and rst7 files

```
> saveamberparm dna1 polyAT_wat.prm7 polyAT_wat.rst7
```

Now, we have all of the input files we need to complete this tutorial and we can progress to running minimization and molecular dynamics simulations.

4.2 Solvation and Simulations

Water plays an integral role in the structure of proteins and nucleic acids and some representation of solvent effects during a simulation is frequently critical. However, when interrogating biological systems, simulations with explicit solvation can become computationally demanding. Biopolymers large molecules and require several thousands of waters to adequately solvate. As such, *in vacuo* simulations are tempting and tricks have been applied to keep the base pairs from fraying, through the addition of Watson-Crick base pair restraints and the reduction of the charges on the phosphate groups. Additionally, newer versions of AMBER contain the generalized Born model for implicit solvation which, though more expensive and *in vacuo*, provides a much better solvent representation than simply using a distance dependent dielectric constant. While a nanosecond or so of *in vacuo* DNA simulation can take only minutes on a 3GHz P4, adding a periodic box of water that surrounds the DNA by roughly 10 angstroms, extends the simulation to several days.
Even with advances in computer power and methodological improvements, such as the application of Ewald methods, which allow routine simulations of nucleic acids with explicit solvent and counter-ions in the nanosecond time range, there is still dependence of the results on the molecular mechanical force field. If you can afford it, include the solvent and the explicit net-neutralizing counterions. Frequently scientists have to balance cost vs. accuracy. To this end, it is important to understand the inadequacies of the force field being used and to pay attention to be aware of their limitations.

4.3 Running Minimization and Molecular Dynamics

We can follow the same protocol that we performed in the previous tutorial to minimize the DNA structure *in vacuo*. The mdin file we want to use can be created by

```
[~]$ vi polyAT_vac.in
```

and include the following text

```
polyA-polyT 10-mer:  initial minimization prior to MD
&cntrl
imin = 1,
maxcyc = 500,
ncyc = 250,
ntb = 0,
igb = 0,
cut = 12
/
```

This file we will be able to run the minimization with the command

```
[~]$ $AMBERHOME/bin/sander -O -i polyAT_vac.in -o polyAT_vac.out -c
polyAT_vac.rst7 -p polyAT_vac.prm7 -r polyAT_vac.ncrst
```

and should run fairly quickly. If you look at the output file, the run should finish with a successful completion message and you should note that the energy of the system should have decreased considerably between the first and last steps.

DELIVERABLE: You should know how to extract minimization energy profiles from these output files. Please do so from this minimization, you will want to save the plot for use later!

4.4 Creating a PDB file from AMBER coordinates

After minimization, you may want to generate a new pdb file so you can look at the structure using the minimized coordinates. A pdb file can be created from the parameter topology and coordinates (rst7 or ncrst) using the program `ambpdb`.

```
[~]$ $AMBERHOME/bin/ambpdb -p polyAT_vac.prm7 -c polyAT_vac.ncrst >
polyAT_vac_min.pdb
```

In general, users should carefully inspect any structure they are using; specifically check to make sure the hydrogens were placed where you thought they should be, histidines are in the correct protonation state, the terminal residues are properly terminated, stereochemistry is reasonable, etc. There is nothing worse than finding out after you've run a nanosecond of solvated dynamics that an H1' atom was on the wrong side and that you have simulated some strange anomer of DNA!

DELIVERABLE: Use VMD to load in your PDB files from before and after minimization. Use the representations of each to color the models differently, and if you have time, play around with the way you can draw the DNA. Com-

ment on what changed (it should be subtle!).

4.5 Running MD *in vacuo*

To make the simulations tractable for this tutorial, short simulations will be run on the order of 100 ps. These simulations would likely be unable to answer specific research questions, but you will see that they are still have a significant cost, depending on what type of machine you using. To begin, we will run simulations *in vacuo* with sander. We will run two different simulations to demonstrate the differences in long-range electrostatics, one simulation with a 12.0 angstrom cutoff and one with no (infinite) cutoff. The CUT variable specifies the cut off range for the long range non-bonded interactions. Here, two different values for the cutoff will be used, one run with a cut off of 12 angstroms (`CUT = 12.0`) and one run without a cutoff. To run without a cutoff we simply set CUT to be larger than the extent of the system (e.g. `CUT = 999`).

For temperature regulation, we will use the Langevin thermostat (`NTT=3`) to maintain the temperature of our system at 300 K. This temperature control method uses Langevin dynamics with a collision frequency given by `GAMMA_LN`. If we have a reasonably good structure, which we do in this case, we can actually start the system at 300 K and avoid the need to slowly heat over, say, 20 ps from 0 K to room temperature. One should use the Langevin temperature regulation scheme with care, however. While it allows efficient equilibration the system's temperature, it will alter the fast dynamics of your system. As such, if you are interested in things such as correlation functions, it is often better to equilibrate your system using `NTT=3` and then once equilibrated, pure Newtonian dynamics (`NTT=0`). It should also be noted that the performance of `NTT=3` is less than a NVT simulation using Berendsen temperature scheme (`NTT=1`) or an NVE (`NTT=0`) simulation. Hence it can often be beneficial to heat in NVT with `NTT=3`, equilibrate in NPT, then run production as either NVE (NTT=0) or as NVT with a weak coupling constant of around 10ps^{-1}. The exact choice, though, will depend on what you aim to learn from your simulation and so you should consult literature for previous examples and discussions.

For simplicity, we will run this entire simulation with `NTT=3` and `GAMMA_LN=1`. We shall also set the initial and final temperatures to 300K which will mean our system's temperature should remain around 300 K. In these two examples we will run a total of 100,000 steps each with a 1fs time step giving simulation lengths of 100 ps (100,000 x 1fs). The two input files are shown below, first `polyAT_vac_mdcut.in`

```
10-mer DNA MD in-vacuo, 12 angstrom cut off
&cntrl
imin = 0, ntb = 0,
igb = 0, ntpr = 100, ntwx = 100,
ntt = 3, gamma_ln = 1.0,
tempi = 300.0, temp0 = 300.0
nstlim = 100000, dt = 0.001,
```

```
cut = 12.0
/
```

and `polyAT_vac_mdnocut.in`

```
10-mer DNA MD in-vacuo, infinite cut off
&cntrl
imin = 0, ntb = 0,
igb = 0, ntpr = 100, ntwx = 100,
ntt = 3, gamma_ln = 1.0,
tempi = 300.0, temp0 = 300.0
nstlim = 100000, dt = 0.001,
cut = 999
/
```

We can run the two simulations by issuing the following commands

```
[~]$ $AMBERHOME/bin/sander -O -i polyAT_vac_mdcut.in -o polyAT_vac_mdcut.out
-c polyAT_vac_min.ncrst -p polyAT_vac.prm7 -r polyAT_vac_mdcut.ncrst
-x polyAT_vac_mdcut.nc
```

and

```
[~]$ $AMBERHOME/bin/sander -O -i polyAT_vac_mdnocut.in -o polyAT_vac_mdnocut.out
-c polyAT_vac_min.ncrst -p polyAT_vac.prm7 -r polyAT_vac_mdnocut.ncrst
-x polyAT_vac_mdnocut.nc
```

Note that since you have a multicore machine, you can run these jobs simultaneously on different processors. These will take longer to run than the minimization, and will take on the order of 5 to 10 minutes. If you would like to 'track' the progress of your simulations, there are a few basic commands that can be useful. In a terminal you can type `top` which will show you which processes are running on your computer. You should be able to see your job running. To exit top, it the 'q' key. Additionally, you can examine the output file by using the command `less`

```
[~]$ less polyAT_vac_mdnocut.out
```

which controls similarly to Vim, but does not have all of the functionality (or the hardware demands). To go to the bottom of the file you can type 'G', which will bring you to the current last line. `less` will update the file as it is written so you can hit 'G' again and it should scroll further.

Once they are completed we can move on to analyzing the results. However, DELIVERABLE: Why would the mdnocut simulations expect to take longer to

simulate than the mdcut simulations? Explain a rationale and also comment on why your simulations were on the same general timescale.

4.6 Analyzing the results

When the simulation stops, we will want to look through the mdout files to make sure the jobs completed successfully. Once you do, you might notice that the "nocut" files crashed and terminated prematurely. Most of the time, if a simulation crashes immediately, there is a problem with the input file. If a simulation runs but crashes after a prolonged time, then the simulation might have reached a point where it cannot continue. There are a wide range of problems that can occur and it is important to analyze crashed jobs to discover what issues lead to the crash to ensure that you do not have systemic problems in the simulation protocols. It is also a good idea to extract the various energies as a function of time to plot the total, kinetic, potential energies, etc. We will also want to calculate RMSD vs. time.

Most of this information can be pulled directly from of the mdout file. We could create our own awk/sed/perl/python scripts to accomplish this task but, luckily, we can use a pre-packaged script; process_mdout.pl . It has been designed to process mdout files and create a series of files with the various different pieces of information in them. The script uses a default output filename so it is best to create a sub-directory for each of your output files and move to there before running the script

```
[~]$ mkdir nocut_analysis cut_analysis
```

and then move into the first directory

```
[~]$ cd cut_analysis
```

and run the analysis script

```
[~]$ $AMBERHOME/bin/process_mdout.perl ../polyAT_vac_mdcut.out
```

and repeat for the "nocut" files even though the simulation may not have completed all simulation steps. This script takes a series of mdout files and will create a series, leading off with the prefix "summary". such as "summary.EPTOT", of output files. You can plot the summary files with your favorite graphing program. A useful program for just quickly looking at a plot of a file is xmgrace, a very simple plotting program, but ideal for our purpose. To plot the total potential energy, of both runs, as a function of time we run, assuming we are in our master simulation directory

```
[~]$ xmgrace ./polyAT_vac_mdcut/summary.EPTOT ./polyAT_vac_mdnocut/
summary.EPTOT
```

You should be able to see that the 12 angstrom cutoff simulation seems to be fairly stable since the potential energy is fluctuating around a constant mean value. The simulation without a cutoff, however, is significantly different. To begin with the potential energy is some 3,000 kcal/mol higher than the 12 angstrom cutoff simulation. The potential energy is so large in fact that it is actually positive. The potential energy also decreases rapidly over the first 10 ps of simulation suggesting that a large structural change is occurring. The simulation then ends abruptly after 26ps with the following error:

```
SANDER BOMB in subroutine Routine:  map_coords (ew_force.f)
Atom out of bounds.  If a restart has been written,
restarting should resolve the error
```

DELIVERABLE: Save plots for the kinetic, potential, and total energies. Why the difference in potential energy; what is the major contribution to the difference? . What happened in the nocut simulations, and why did it happen? At this point, conjecture about what might be going on.

4.7 Calculating the RMSD

The next step in analyzing our results is to calculate the RMSD as a function of time using cpptraj (an analysis program provided with AmberTools). The precise parameter we will be calculating in this example is the mass weighted RMSD fit between each successive structure and the first structure of our trajectory. This is done by providing an input file to cpptraj containing a list of commands describing the relevant files and what we want it to do, etc. So we will make two input files `polyAT_vac_mdcut.cppin` and `polyAT_vac_mdnocut.cppin` with input commands for cpptraj like:

```
trajin polyAT_vac_mdcut.nc
rms first mass out polyAT_vac_mdcut.rms time 0.1
```

and then run them with the commands

```
[~]$ $AMBERHOME/bin/cpptraj -p polyAT_vac.prm7 -i polyAT_vac_mdcut.cppin
[~]$ $AMBERHOME/bin/cpptraj -p polyAT_vac.prm7 -i polyAT_vac_mdnocut.cppin
```

where trajin specifies the name of the trajectory file to process, rms first mass tells cpptraj that we want it to calculate a *mass weighted* RMS fit to the *first* structure. out specifies the name of the file to write the results to and time 0.1 tells cpptraj that each frame of the coordinate file represents 0.1 ps. We should get two output files, one for each simulation, and can now plot the RMSD's vs time. Note, RMSD is in angstroms and time is in ps

```
[~]$ xmgrace polyAT_vac_mdcut.rms polyAT_vac_mdnocut.rms
```

From these plots, we should be able to see the problem. While the 12 angstrom cutoff simulation has a largely constant RMSD around 2.2 angstroms the no cutoff simulation shows a steadily increasing RMSD which after about 20 ps is already over 30 angstroms! This would suggest that something has gone wrong with the simulation since our system has essentially "blown up".

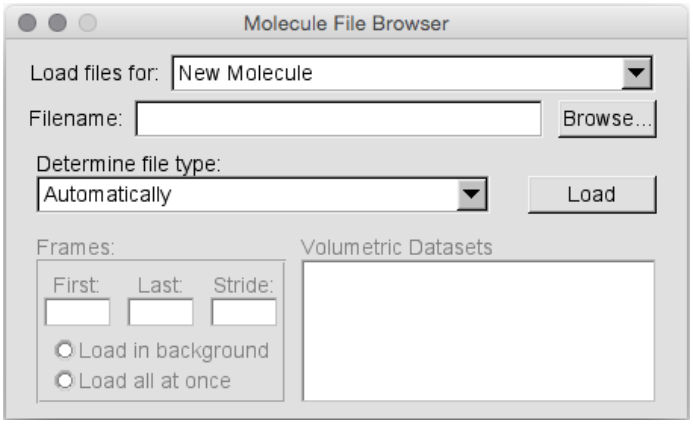
DELIVERABLE: Is not having the cutoff problematic or does this result suggest a larger problem with running a simulation in vacuum? Why does the use of a cutoff result in a stable trajectory while the use of no cutoff, which should in theory be more accurate, leads to catastrophic results? Address the postulations you had before on what happened in the simulations with this new information.

4.8 Visualizing the trajectories with VMD

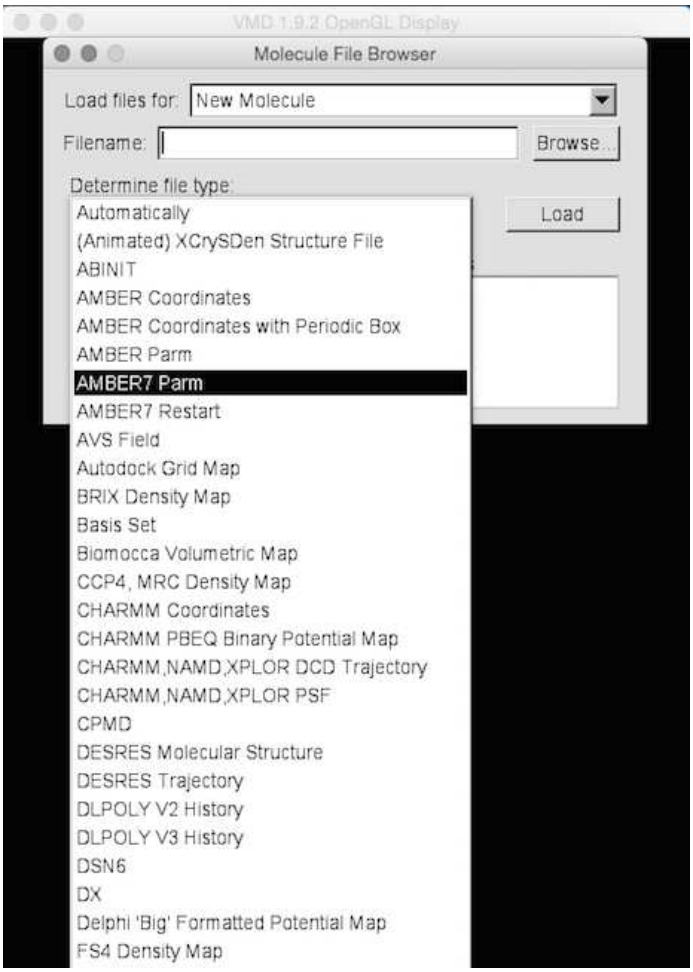
Our DNA molecule consists of two chains held together by hydrogen bonds. Let's take a look at it in vmd:

```
[~]$ vmd
```

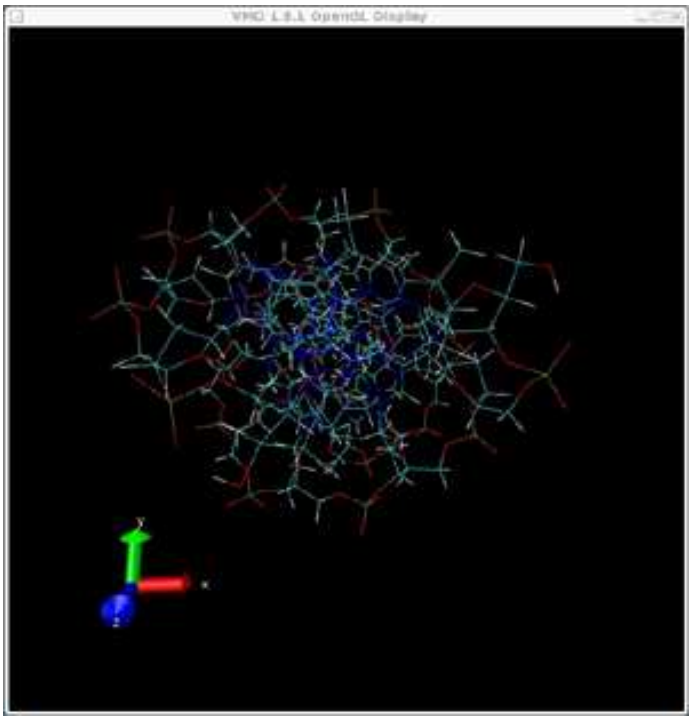
Open our 12 angstrom cutoff trajectory S. file. Select `File` → `New Molecule`



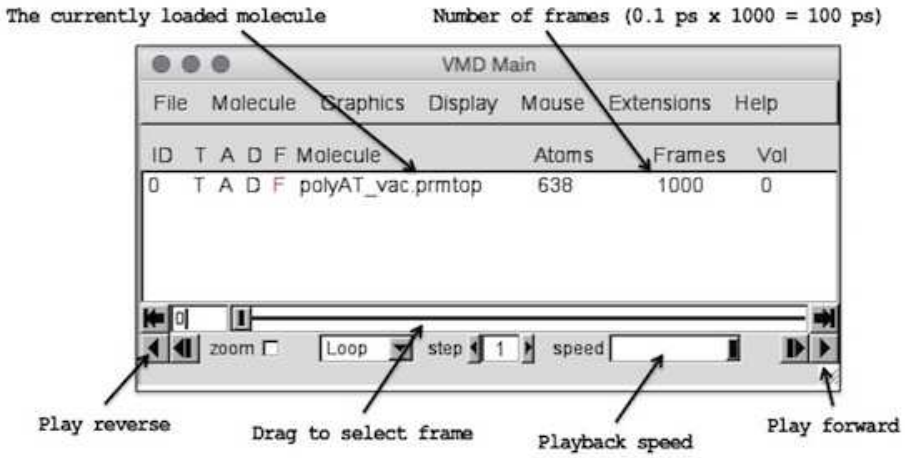
VMD supports multiple trajectory files for a single molecule. Thus a molecule is defined by loading the associated parameter/topology file. To start, select `Browse` and find the *polyAT_vac.prm7* file. Then under the heading 'Determine file type:' select AMBER7 Parm.



and click `Load`. Next, we need to choose which trajectory to load. In this case, we will load the 12 angstrom cutoff trajectory *polyAT_vac_cut.nc* file. Select `Browse` and find the file. Then, under 'Determine file type' select AMBER Coordinates. Note: if this were a periodic boundary simulation then we would select "AMBER Coordinates with Periodic Box" since the trajectory file would also contain information on the box size. In this situation, however, there is no box information since this is a non-periodic *in vacuo* simulation. Hit `Load` again you should see all of the frames loaded into the main molecule window with 1,000 frames in total.



Use the playback tools in the VMD main control panel to play our movie



Play the trajectory movie and notice how the DNA holds its secondary structure. Though there is considerable movement in the extremities, the overall structure is preserved. Lets take a look at the trajectory obtained from our no cutoff

simulation. We can load a new trajectory with `File` → `New Molecule` again with the *polyAT_vac.prm7* file but load the *polyAT_vac_mdnocut.nc* trajectory with it. MD should load the trajectory up until the point where the simulation crashed. Have a look at the trajectory, the difference from the last simulation should be obvious. The instability of the DNA dimer is clear.
DELIVERABLE: Is this simulation representative of reality? A stable trajectory doesn't necessarily mean it is correct. Is a strand of charged DNA in vacuum really likely to be stable? With the information from visualizing the trajectories (and possibly measuring some distances in VMD with Mouse→Label→Bonds) ask what could be the differences in the cut and nocut simulations that lead to the behaviour observed?
One way to improve considerably on our *in vacuo* simulations is to make our physical model of DNA closer to reality. This is the subject of the next stages of our tutorial.

4.9 Running Minimization and MD in implicit solvent

In this example we shall re-run our in vacuo simulations using the standard pairwise Born solvation model. The first stage is to minimize our initial system. We use the same prm7 and rst7 files as previously (polyAT_vac.prm7, polyAT_vac.rst7). This time around though we will modify our input file to turn on the generalized Born method (IGB=1). We can make a new input file polyAT_gbmin.in

```
polyA-polyT 10-mer:  initial minimization prior to MD GB model
&cntrl
imin = 1,
maxcyc = 500,
ncyc = 250,
ntb = 0,
igb = 1,
cut = 12
/
```

and run the minimization with

```
[~]$ $AMBERHOME/bin/sander -O -i polyAT_gbmin.in -o polyAT_gbmin.out -c polyAT_
vac.rst7 -p polyAT_vac.prm7 -r polyAT_gbmin.ncrst
```

The first thing you should notice is that this takes considerably longer than the *in vacuo* minimization. Herein lies the problem with including solvent in simulations. As we will see, it is often an expense that cannot be avoided. Lets take a quick look at the output file produced during the minimization and you will see that the energy has dropped between the first and last steps however it is worth noting that the change as compared to the *in vacuo* minimization.
DELIVERABLE: Plot the minimization energy of the implicit solvent calcula-

tion. Compare the magnitude of change between the solvated and unsolvated systems. What could this finding relate about the stability of DNA in these different environments?
Again we can create PDB files to compare structures if you would like to visualize them in VMD

```
[~]$ $AMBERHOME/bin/ambpdb -p polyAT_vac.prm7 -c polyAT_gbmin.ncrst > polyAT_
gbmin.pdb
```

Now we can run molecular dynamics of our system. We are using the same settings as before, namely we turn off minimization (IMIN=0). We disable periodicity (NTB=0) but this time set IGB=1 since we want to use the Born implicit solvent model. We will again write information to the output file and trajectory coordinates file every 100 steps (NTPR=100, NTWX=100). Two different values for the cutoff will be used, one run will be with a cutoff of 12 angstroms (CUT = 12.0) and one run will be without a cutoff (CUT = 999). For temperature regulation we will use the Langevin thermostat (NTT=3, GAMMA_LN=1) to maintain the temperature of our system at 300 K. Creating the input file *polyAT_gbcut.in*

```
10-mer DNA MD Generalized Born, 12 angstrom cut off
&cntrl
imin = 0, ntb = 0,
igb = 1, ntpr = 100, ntwx = 100,
ntt = 3, gamma_ln = 1.0,
tempi = 300.0, temp0 = 300.0
nstlim = 100000, dt = 0.001,
cut = 12.0
/
```

and input file *polyAT_gbnocut.in*

```
10-mer DNA MD Generalized Born, infinite cut off
&cntrl
imin = 0, ntb = 0,
igb = 1, ntpr = 100, ntwx = 100,
ntt = 3, gamma_ln = 1.0,
tempi = 300.0, temp0 = 300.0
nstlim = 100000, dt = 0.001,
cut = 999
/
```

and can run these jobs with the following commands

```
[~]$ $AMBERHOME/bin/sander -O -i polyAT_gbcut.in -o polyAT_gbcut.out -c polyAT_
gbmin.ncrst -p polyAT_vac.prm7 -r polyAT_gbcut.ncrst -x polyAT_gbcut.
nc [~]$ $AMBERHOME/bin/sander -O -i polyAT_gbnocut.in -o polyAT_gbnocut.
out -c polyAT_gbmin.ncrst -p polyAT_vac.prm7 -r polyAT_gbnocut.ncrst -x polyAT_
gbnocut.nc
```

These will take considerably longer to run than in vacuo MD, so you might want to get to this point and leave. They will take on the order of **one hour**.

4.10 Analyzing the results

Again, we will use the pre-packaged analysis script and follow the same protocols as we did earlier. We will want to make new analysis directories

```
[~]$ mkdir gbcut_analysis gbnocut_analysis
[~]$ cd gbcut_analysis
[~]$ $AMBERHOME/bin/process_mdout.pl ../polyAT_gbcut.out
[~]$ cd ../gbnocut_analysis
[~]$ $AMBERHOME/bin/process_mdout.pl ../polyAT_gbnocut.out
[~]$ cd ../
[~]$ xmgrace gbcut_analysis/summary.EPTOT gbnocut_analysis/summary.
EPTOT
```

The first thing you should notice is that the plots (cut and nocut) are now very similar. The cut off value has made much less of a difference than it did *in vacuo*.
DELIVERABLE: Why would the simulations be more similar between the gb simulations as compared to the vacuum simulations? What physical considerations are being dealt with when using the implicit solvation model? Please include the plots with discussion. Comment on the differences in fluctuations of the kinetic and potential energies.
Lets see what difference there is in the RMSD. Make a cpptraj input files *polyAT_gbcut.cppin*

```
trajin polyAT_gbcut.nc
rms first mass out polyAT_gbcut.rms time 0.1
```

and *polyAT_gbnocut.cppin*

```
trajin polyAT_gbnocut.nc
rms first mass out polyAT_gbnocut.rms time 0.1
```

then run the calculation with cpptraj

```
[~]$ $AMBERHOME/bin/cpptraj -p polyAT_vac.prm7 -i polyAT_gbcut.cppin
[~]$ $AMBERHOME/bin/cpptraj -p polyAT_vac.prm7 -i polyAT_gbnocut.cppin
```

and finally plot

```
[~]$ xmgrace polyAT_gbcut.rms polyAT_gbnocut.rms
```

Notice how both simulations are much more stable than the vacuum case. It does, however, make a big difference to the time required for the simulation.
DELIVERABLE: Plot the RMSD and include it with the other assigned information.
If you care to, you can also examine the simulation trajectory in VMD as we did before. Also, if you got this far and want some fun, thought provoking, extra steps to look at for your simulations here are two *optional* (or possibly optionally assigned) objectives that you can attempt:

- Perform another analysis. Only this time, use the explicit simulation DNA files found in the ref directory. Look at things like the energies, the RMSD, and the total simulation times (while not run on the exact some computer, they should still be representative of the difference in simulation times). Why would you want to run implicit solvent simulations? What are the advantages of running explicit solvent simulations?

- If you have *a lot* of extra time, run the explicit solvent calculations for yourself. Only this time, increase the temperature of the simulation to 350K. Upon completion, perform the analysis and look at the RMSD. Comment on the differences you see.

5 Project

5.1 Introduction

Now that you have worked through setting up simulations for both protein peptides and of nucleic acids in a variety of conditions you should be ready to start exploring molecular dynamics simulations related to your own research interests. While we have not discussed all of the powerful techniques available to veteran users, you should have an understanding of how to set-up and run simulations as well as an idea of some of the common pitfalls that might come up along the way. For real-world applications of MD you would likely employ more advanced simulation protocols to enhance local sampling of important areas of phase space or alter the nature of the Hamiltonian (potential energy landscape) that you are sampling to obtain a better understanding of the free energies working upon your biochemical systems of interest.

For this final assignment we will perform a simple molecular dynamics simulation, much like those that you have performed previously in the tutorials. You will use the CaN.pdb file (the structure of a bound peptide) to run two different simulations, one in explicit tip3p water and Cl^- ions and another in a General Born implicit solvent model.

5.2 Background

Calmodulin (CaM) is a ubiquitous protein that plays a key role in calcium-mediated signal transduction. It has been shown that CaM binds and regulates more than 300 target proteins and that its structural plasticity is crucial for enabling its interaction with the diverse partners (Ikura and Ames, 2006). CaM consists of two homologous domains, the N-terminal domain (NTD) and the C-terminal domain (CTD), which are separated by an interdomain linker. Each domain is composed of two EF-hand helix-loop-helix motifs. These motifs occupy a "closed" conformation in the calcium-free (apo-CaM) state, in which the helices in the two pairs of EF hands are closely packed together. Ca^{2+} ligation during a calcium spike leads to an "open" state (Ca^{2+}-CaM) in which significant changes in conformation in each EF-hand pair result in the exposure of a hydrophobic cleft in both domains (Zhang et al., 1995; Ikura, 1996). The exposure of this cleft increases the affinity of CaM for a wide range of binding partners (Osawa et al., 1998; Crivici and Ikura, 1995; Bayley et al., 1996).

One of the binding partners is calcineurin (CaN), a serine/threonine phosphatase. CaN has, among other elements, a catalytic and regulatory domain. The regulatory domain contains an autoinhibitory region and a CaM-binding regions. At low calcium concentrations, CaN is in an inactive state, with its autoinhibitory region occluding the active-site cleft of the catalytic domain. As a result of an increase in calcium concentration, CaM binds four calcium ions, which changes the structure of CaM and enables binding of the CaM-binding

region in CaN. CaM binding releases autoinhibition and activates CaN. Experiments have shown that, like many other CaM-binding regions, the one of CaN is in an intrinsically disordered protein part. Thus, it lacks a tertiary structure before partner binding. Upon CaM binding, the region folds into a helix.

The goal of the assignment is to compare the dynamics of the CaN binding partner in its unbound state in both implicit and explicit solvent models, specifically to interrogate how the dynamics of this small peptide are affected by the choice of solvent model. Key resources to use which can facilitate this enquiry could be:

- To use the process_mdout.perl script to look at energies and how they change over the simulation.
- To examine how the secondary structures of the two simulations differ. To do this in VMD use `Extensions`→`Analysis`→`Timeline` then `Calculate`→`Calc. Sec. Struct`. Information of the Timeline extension can be found here.
- The RMSD from your starting structure of CaN over the simulation, barring in mind that the initial structure is that of the bound peptide. Does this differ between the solvent models?
- Comparing the radius of gyration of the peptide over the simulation between the two different solvent models. You can do this analysis quickly in cpptraj (much like the RMSD) by loading in the parameter and the trajectory files then issuing the command `radgyr out <output_filname.rgy>`.
- Looking at the Contact Map in VMD under `Extensions`→`Analysis`→`Contact Map` then `Calculate`→`Calc. Res-Res Dist.`. Repeat this calculation for different windows in the simulation and compare.

Some special consideration should be placed into the questions:

- To what extent do your simulations agree with the experimental observation that the CaM-binding region in CaN is intrinsically disordered and folds upon binding with CaM? If not why? Which solvation model is giving a better representation of reality?
- Is the force field used adequate for the simulation? The ff14SB is designed particularly well for general simulations, however these parameters might not be adequate for proteins with high intrinsic disorder.
- How does the computational efficiency of these simulations change with the solvent model? How important is model accuracy vs. sampling time? Under what cases would an implicit or explicit solvent model be preferred?

5.3 Simulation parameters

You have learned how to setup and run appropriate simulations to make the comparisons asked for in this project, however, you might want to think critically about the simulations before you start them. What conditions should be applied to the minimization, heating, and production steps. Furthermore, we want to run *very long* simulations (comparatively) for this project. Ideally, you will have a total simulation time of 40ns for your implicit solvent simulations, for instance. It is therefore recommended to use a 2fs timestep for your dynamics propagation. Also, in order to not make your files unmanageable, you would also want to adjust how frequently you print information to your *.nc and *.out files.

5.4 Deliverables

Report your findings to me (erich.kuechler@msl.ubc.ca) as you did for the previous tutorials. The questions above are *suggestions* and do not need to be followed directly or in that order. Additionally, you are welcome to add in other information (specific notes on visualizations of the trajectory for instance) that you think are important. To screen capture a specific window: make sure the window you want to capture is the active window them hit `Atl+Prt Scr`. After a few seconds a pop-up should appear to save an image.