10.637: Lab3 (GL1): Classical molecular dynamics **10/03/23**

Submitting your lab report: Unlike *Labs 1* and *2*, this lab will be a graded lab (GL). Your lab report should address the prompts included at the end of this handout. It is due by **11:59 PM EDT on (10/19/23)**. The GL1 homework assignment is at the end of this handout. Submit your report using Canvas on the "assignments" tab as a single .pdf file formatted as "FirstName_LastName_GL1.pdf"

In today's lab, we will use the command line for the first time to run classical molecular dynamics (MD) with AMBER on water. *These same tools are used most commonly to carry out biochemical simulations of proteins or solutions.* If you're not comfortable with the command line yet, don't worry. We'll review everything you need.

Using this document:

1) You should type the commands because copying and pasting directly from the PDF may lead to some errant characters. It is also good practice to get used to typing the commands we need to use in this class.
2) To help you follow along with the tutorial, we will show all commands and input files in `courier font like this`. However, we will show the commands we need you to type in **`green and bold courier font like this`**. Optional commands will not be colored green.
3) If you see angled brackets without additional spaces such as <Your Username>, you should replace everything (including the brackets) with the specified prompt. Optional arguments are provided in square brackets – e.g. [out <outfile>] means that you can specify an output file with the name provided for <outfile>.
4) *Comments inside input files to explain them in the document will be prefaced with a "!" but will not appear in your local copy of the file.*
5) Text in **red** indicates that you will need to make an action. Text in **blue** indicates optional notes or hints for if you get stuck.

Today, we will be SSHing into Expanse to run the calculations.

Quick login reminder:

```
ssh <ACCESS ID>@login.expanse.sdsc.edu
```

Quick overview of running jobs on ACCESS:

You can submit a job with

```
sbatch jobscript
```

or

```
sbatch jobscript --reservation=ITM101RES
```

Note: This will only work from 11am-1pm during 10/3

You can check the status of your jobs with

```
squeue -u <ACCESS ID>
```

You can delete a job with

```
scancel <job #>
```

To watch the output file of a run as it is going:

```
tail -f <filename>
```

NOTE BEFORE YOU START: Sections A and B require intensive compute resources and should be prioritized during the lab period. Section C will run quickly and can be carried out while you are waiting for parts A and B to finish. Section D can be done at home.

A. Simulating boxes of water.

We will test out some water models that we discussed in class (TIP3P and TIP4PFB). If you do not remember what these force fields refer to, please review the notes from *Topic 3*. In this first section, we have built the water boxes for you, enabling you to skip the preparation you'd normally do using the **`tleap`** utility in AMBER.

Using these starting water boxes, we will geometry optimize, heat, equilibrate, and run production dynamics on these water boxes. Once this is done, we will carry out analysis on the water simulation production run. Our main goal is to understand differences in observables for the two different water models.

1) First, login to Expanse.
   ```
   ssh <ACCESS ID>@login.expanse.sdsc.edu
   ```

2) Make a directory for the files related to this lab and then go into the directory.
   ```
   mkdir lab3
   cd lab3
   ```
3) Now copy the pre-built water boxes into your directory:
   ```
   cp -pr /expanse/lustre/projects/itm101/hkulik/lab3/* ./
   ```
   Note: This command will copy over three directories, one for the water boxes (Section A), one for the salt solution (Section B), and one for the charges of water and small molecules (Section C). For each water box, there is a coordinate file (**`.inpcrd`**), a topology file (**`.prmtop`**), a series of AMBER input files (**`.in`**), two jobscripts (**`.q`**).

4) To start on Section A, for example, you can see these files for the TIP3P water model by changing directories:
   ```
   cd water/TIP3P
   ```
5) List the files using:
   ```
   ls
   ```
   Note: These water boxes were constructed using **`tleap`**, which does not do a good job of setting up the initial density of our water. We are interested in room temperature and ambient pressure water that has a 1 g/cm^3 density, but the initial densities of the structures generated by **tleap** are both about 0.7 g/cm^3. This is a good example of why it's important to equilibrate a structure prior to a molecular dynamics production run. The equilibration is accomplished by **`amber_equil.q`** and consists of 4 steps I-IV:
   I. **2,000** steps of unrestrained minimization (**`unrestrained_min.in`**)
   II. **10 ps** of quick NVT heating (**`constV_equil.in`**)
   III. **7.5 ps** of quick NPT equilibration (**`constP_equil.in`**)
   IV. **500 ps** of longer NPT equilibration (**`constP_run.in`**)

   This equilibration should take about 5 minutes for each water box. Make sure to get it set up quickly so you can complete the production run during class.
   After equilibration, we will run a **5 ns** production NPT simulation **`constP_production.in`** using **`amber_prod.q.`**

   Take a moment to view each of the input files with:
   ```
   nano <file name>
   ```
   Hint: You can also "tab complete" by typing the first few letters of a file then pressing tab. Either one unique filename will appear or you will get a list of all the files that have those

same few characters. This works with wildcards as well.

6) Below are the contents of these files with a little more explanation. In some cases, you need to fill in a few blanks (currently marked with **??**). Use **`nano`** to make these changes. Remember to use *Ctrl+X* to save your changes and exit **`nano`** when you are done.

The first file, **`unrestrained_min.in`** carries out an initial geometry optimization on our water box.

unrestrained_min.in (no changes):

```
&cntrl              ! Control namelist
 imin = 1,          ! Minimization is on
 maxcyc = 2000,     ! Number of cycles to run minimization
 ncyc = 2000,       ! Number of cycles to run steepest descent
 ntb = 1,           ! Constant volume PBCs
 cut = 9.0,         ! Nonbonded cutoff distance in Å-small because our cell is small.
/
```

For this next file, we need to adjust the number of steps (**nstlim**) and time per step (**dt**) such that the total equilibration time is **10 picoseconds**. Use a time step of **2 femtoseconds** but note that you will have to specify this time in picoseconds in the input file. Calculate the number of dynamics steps based on this time per second and the total simulation time.

constV_equil.in (changes indicated as ?? in red):

```
&cntrl              ! Control namelist
 nstlim=??,         ! Number of dynamics steps**
 dt=??,             ! Timestep in ps**
 ntx=1,             ! Read in the coordinates without velocities.
 irest=0,           ! We are not restarting the simulation
 ntpr=500,          ! How frequently we print progress
 ntwr=5000,         ! How frequently we write the restart file
 ntwx=5000,         ! How frequently we write the mdcrd file
 tempi=100.0,       ! Initial temperature
 temp0=298.15,      ! Target temperature- to compare to experiment
 ntt=3,             ! Langevin thermostat
 gamma_ln=1.0,      ! Friction constant for Langevin.
 ig=-1,             ! Random seed for starting velocities.
 ntb=1,             ! Constant volume PBCs
 ntc=2,             ! SHAKE algorithm to constrain bonds with hydrogen**
 ntf=2,             ! Interactions involving H-atoms omitted from potential function**
 cut = 9.0,         ! Nonbonded cutoff distance in Å-small because our cell is small.
 ntxo=1,            ! Make restart files write in plain text instead of a binary format
 ioutfm=1           ! Controls format of trajectory files
/
```

The **`constP_equil.in`** is similar to **`constV_equil.in`** except for the following changes that you need to make:

- We are restarting the MD so we need to use **irest=1** and **ntx=5** for coordinates and velocities read in.
- NPT: **ntb=2** for constant pressure periodic boundary conditions (PBCs) and **ntp = 1** for constant pressure dynamics.

constP_equil.in (changes indicated as ?? in red):

```
&cntrl          ! Control namelist
 nstlim=30000,  ! Number of dynamics steps**
 dt=0.0005,     ! Timestep in ps**
 ntx=??,        ! Adjust this variable to restart the MD based on previous MD
 irest=??,      ! Adjust this variable to read in the coordinates and velocities
 ntpr=500,      ! How frequently we print progress
 ntwr=5000,     ! How frequently we write the restart file
 ntwx=5000,     ! How frequently we write the mdcrd file
 temp0=298.15,  ! Target temperature- to compare to experiment
 ntt=3,         ! Langevin thermostat
 gamma_ln=1.0,  ! Friction constant for Langevin.
 ig=-1,         ! Random seed for starting velocities.
 ntb=??,        ! Adjust this variable to change to constant pressure PBCs
 ntp=??,        ! Adjust this variable to specify the NPT ensemble (instead of NVT)
 ntc=2,         ! SHAKE algorithm to constrain bonds with hydrogen**
 ntf=2,         ! Interactions involving H-atoms omitted from potential function**
 cut=9.0,       ! Nonbonded cutoff distance in Å-small because our cell is small.
 ntxo=1,        ! Make restart files write in plain text instead of a binary format
 ioutfm=1       ! Controls format of trajectory files
/
```

No changes for **`constP_run.in`**, which is similar to **`constP_equil.in`** except:

- Rapid fluctuations can happen once we allow the cell size to adjust. To be cautious, we added 30000 timesteps (**nstlim=30000**) with a ½ fs timestep (**dt=0.0005**) in **`constP_equil.in`**. The goal was to ensure numerical stability, which is especially important for the GPU version of the code we used.
- Here, we increase the timestep to 2 fs (**dt=0.002**) and carry out a longer 1 ns NPT equilibration run (**nstlim=500000**).

7) Now we are ready to submit our equilibration job. Take a look at **`amber_equil.q`** using **`nano`**. We recommend changing the name of the job to specify your username and the type of water model used in this run. Notice that there is a series of four AMBER runs, where each subsequent run is initialized using the "**.restart**" file from the run before it. All AMBER commands follow the following general format (**Note:** these commands are typed on one line but we have put them on two for readability here):

```
<amber executable> -O -i <input> -o <output> -p <topology> -c <coordinates> -r
<restart out> -x <coordinates out for md>
```

which is repeated several times in the **`amber_equil.q`** job script below.

```
! A header for the queueing system, which is in this case SLURM:
#!/bin/bash                         ! Shell environment, leave this.
#SBATCH -J watermd                  ! Name of the job – revise to make it easier to track
#SBATCH -o equil.out                ! File to put errors into
#SBATCH --partition=gpu-shared      ! Use the shared GPU queue
#SBATCH -t 1:00:00                  ! Time limit in HH:MM:SS, max is 48 hrs
#SBATCH --gres=gpu:1                ! Use a single GPU
#SBATCH --ntasks-per-node=1         ! Use a thread for our 1 GPU
#SBATCH -A itm101                   ! Name of the account you'll use.

! Commands to initialize our environment:
module load gpu/0.15.4              ! Loading GPU tools (including CUDA)
module load ompenmpi/4.0.4          ! Loading parallelization tools (Amber requires)
module load amber/20                ! Loading Amber including pmemd.cuda

! Amber commands:
prmtop=`/bin/ls *prmtop`            ! Looks for any prmtop file
inpcrd=`/bin/ls *inpcrd`            ! Looks for inpcrd
! Run the unconstrained minimization
echo "Beginning unrestrained minimization..."
pmemd.cuda -O -i unrestrained_min.in -o unrestrained_min.out -p $prmtop  -c $inpcrd -r
unrestrained.restart
! Slowly increase the temperature
echo "Beginning constant volume equilibration..."
pmemd.cuda -O -i constV_equil.in -o constV_equil.out -p $prmtop  -c unrestrained.restart
-r constv.restart -x constv.mdcrd
! NpT equilibration small time step
echo "Beginning constant pressure equilibration..."
pmemd.cuda -O -i constP_equil.in -o constP_equil.out -p $prmtop  -c constv.restart -r
constpeq.restart -x constpeq.mdcrd
!NpT equilibration larger time step
echo "Beginning constant pressure run..."
pmemd.cuda -O -i constP_run.in -o constP_run.out -p $prmtop  -c constpeq.restart -r
constprun.restart -x constprun.mdcrd
```

8) Now you're ready to submit the queue script, which you do with the `sbatch` command. **Note:** To ensure that everyone can complete the calculations during the lab period, we have reserved a set of nodes for the class timeslot – this means your jobs shouldn't have to wait in the queue as long as they normally would. **You can submit your equilibration jobs using the following:**

```
sbatch amber_equil.q --reservation=ITM101RES
```

Note: This will only work during class period. Any jobs still running at 1pm on 10/03 will be killed.

Otherwise use:

```
sbatch amber_equil.q
```

9) Immediately after submitting your job, check to see if it's in the queue with:
   ```
   squeue -u <ACCESS ID>
   ```
   The output of this command should include a job with the name you specified during step 5. It will either be running (it will have the letter "R" next to it) or waiting to run. Once this job finishes (OR if an error occurs in your job setup), it will no longer appear in this output. **Note:** It's important that the equilibration is set up properly and works before we follow up to subsequent steps. That's why we've split the equilibration and production run steps. Feel free to skip to later in this section to prepare your production run input files while the equilibration jobs run**, but do not submit the production script yet**!

10) Check the progress of your jobs with:
    ```
    tail <filename>
    ```
    To watch this update continuously, try:
    ```
    tail -f <filename>
    ```

    When the final line of `constP_run.out` looks like "Total wall time: XX seconds YY hours," the equilibration run is finished. And you can move on to the production run. **Note:** After you finish TIP3P water, you should **repeat steps 3-7** for TIP4PFB water (you will need to edit the files for TIP4PFB, too). If you are waiting for your TIP3P equilibration run to finish, this is an excellent time to prepare and submit your TIP4PFB equilibration run. You can switch to TIP4PFB with:
    ```
    cd ../TIP4PFB
    ```
    or
    ```
    cd ~/lab3/water/TIP4PFB
    ```

11) Once your TIP3P equilibration run is finished, go back to the TIP3P folder (i.e., `cd ../TIP3P)` and submit your production run with:
    ```
    sbatch amber_prod.q
    ```
    For our production run we will be doing a relatively short **1 ns** simulation for the sake of getting the assignments finished during lab. Standard production runs for more complex systems would be carried out for longer times (e.g., 100 ns – 1μs). The output of the production run is written to `constP_production.out`.
    You can check on its progress with: `tail -f constP_production.out`
    This should also take about 5 minutes to complete.

12) When you get the production job submitted, go back and **repeat steps 3-7** for TIP4PFB water (if you have not already), **wait** for it to complete in step 8, then **complete step 9** for the TIP4PFB water system (i.e., submitting the production job). As a reminder, you can switch to the TIP4PFB folder with:
    ```
    cd ../TIP4PFB
    ```
    or
    ```
    cd ~/lab3/water/TIP4PFB
    ```

B. Salt solutions. We'd like to simulate Mg^{2+} in water to understand how well water models can describe the hydration shell around Mg^{2+}. It's well known that Mg^{2+} will form a tight octahedral solvation shell with six coordinating waters that do not exchange on the sub-µs timescale. We're also going to need to neutralize our simulation cell with Cl^- counter-ions. We'd like to know if the TIP3P water model can reproduce the correct Mg^{2+} coordination, so we will simulate 1 Mg and 2Cl in a TIP3P water box. Rather than start from what we know will likely be a poorly equilibrated box of water, let's use an **equilibrated** box of water from *Section A*. We'll use the **`constprun.restart`** from your **TIP3P** run as a starting point.

Note: *If you are short on time*, you may skip steps 3-13 and use the provided files in the folder `/expanse/lustre/projects/itm101/hkulik/lab3_mg2_extra`

1) Previously, we gave you the **`prmtop`** and **`inpcrd`** starting points for the water simulations. This time you are making your own to simulate Mg^{2+} and Cl^- in TIP3P water so you will also need to learn some **`cpptraj`** commands. First, go to the directory of this task:
```
cd ~/lab3/mg2
```

2) Copy over your edited input files from TIP3P (Note that we are using relative path here where "../" refers to the upper-level folder):
```
cp -pr ../water/TIP3P/*.in ./
```

3) Copy over your equilibrated TIP3P water box and its **`prmtop`**
```
cp ../water/TIP3P/constprun.restart ./
cp ../water/TIP3P/tip3pwater.prmtop ./
```

4) Now we need to generate a **pdb** file so we can insert an Mg^{2+} ion into our equilibrated water box PDB, tell **tleap** how big the box is, then neutralize with Cl^- ions. In order to do this, we will need **AmberTools**, which we can access by loading the module with:
```
module load cpu/0.15.4 gcc/9.2.0 openmpi/3.1.6 amber/20
```

5) Open the **`cpptraj`** tool using
```
cpptraj -p tip3pwater.prmtop
```
Note: Refer to the "**cpptraj cheatsheet**" handout on Canvas for more information about this tool

6) This will bring you to another command line type interface. Enter the following commands line-by-line (press enter after each command).
```
trajin constprun.restart
trajout waterbox.pdb
go
quit
```
You should be able to see a file named **`waterbox.pdb`** in the directory.

7) **IMPORTANT:** Delete the tip3pwater.prmtop because you no longer need it**.**
```
rm tip3pwater.prmtop
```

8) Use **`nano`** to read **`mg2+.pdb`** and copy the one line in the file. You should see something

like this:
```
HETATM 0 MG MG 0 0.000 0.000 0.000 1.00 31.08 MG
```

9) Copy your newly created **`waterbox.pdb`** into a new file named **`mgtip3pequil.pdb`** and paste the Mg2+ line into **`mgtip3pequil.pdb`** on the second line, below the information about the periodicity of the box. Be careful to insert a blank line first. PDB files are very whitespace sensitive. On the blank line you created (after the HETATM line) add a **TER** statement after you insert the Mg2+ line as follows:

mgtip3pequil.pdb
```
CRYST1   31.408   30.988   31.261  90.00  90.00  90.00            1
HETATM 0 MG MG 0 0.000 0.000 0.000 1.00 31.08 MG
TER
ATOM      1  O    WAT     1      57.557 -16.531  52.828  1.00  0.00         O
ATOM      2  H1   WAT     1      57.124 -15.815  53.293  1.00  0.00         H
```

Note: We bolded and colored green the two new lines that should be added in the example above. Save your final **`pdb`** file as **`mgtip3pequil.pdb`**.

Now, we're ready to process our **`pdb`** file in **`tleap`** with the following input file, **`leap.in`** (changes required indicated in **red**, comments are in **blue**)

leap.in
```
source oldff/leaprc.ff14SB.redq # for main force field params, solvents.lib, etc
source leaprc.protein.ff14SB # ff14SB has two associated files
mg2=loadamberprep mg2+.prepin # Custom Mg2+ parameters
loadamberparams frcmod.ionsjc_tip3p # TIP3P-optimized ions (for Cl-)
loadamberparams mg2+.frcmod # Custom Mg2+ force field
loadamberparams frcmod.ff14SB
wat=loadpdb mgtip3pequil.pdb # The Mg2+ + water box that you made previously
set wat box {X Y Z} # These are the numbers you got from the instructions below
addIonsRand wat Cl- 0 8 # Add Cl- ions to neutralize the cell (the 0)
check wat
saveamberparm wat mgcl2tip3p.prmtop mgcl2tip3p.inpcrd
savepdb wat mgcl2tip3p.pdb
```

10) To fill in the X Y Z above, issue the command:
```
tail -1 constprun.restart
```
You should see something like: `"31.3726700 30.9534247 31.2266786 90.0000000 90.0000000 90.0000000"` These are the X, Y, and Z box dimensions in Å followed by the angles of the box. Insert the X, Y, and Z coordinates in your **`leap.in`** file.
Note: Leave the braces around the coordinates in **`leap.in`**
Advanced tip: another useful **`tleap`** command you can optionally add to the **`leap.in`** file: **`check wat`**. This tells you if there are any errors, e.g., force field parameters you forgot to load.

11) To make sure AMBER knows where to find force field files be sure to also do:
```
module load cpu/0.15.4 gcc/9.2.0 openmpi/3.1.6 amber/20
```
Note: this line helps loading the CPU prerequisites for **`amber/20`**.

12) Once you have updated your **`leap.in`** you can generate your force field parameters with:

```
tleap -f leap.in
```

13) When you get to the end, check that everything looks like it worked and read through the output. Then you can type `quit` to exit.
    Note: Once you've prepped your **`prmtop`**/**`inpcrd`** files, make sure they are the only **`prmtop/inpcrd`** in the directory (`ls *prmtop` and `ls *inpcrd` to check).

14) You already have the input files, and we've provided you with a combined equilibration and production script called **`amber_all.q`**. When this is all set up, do:
    ```
    sbatch amber_all.q
    ```

C. Typical charges in water and molecular simulations. In this final calculation section, we will compare the charges used in the TIP3P and TIP4PFB water models to calculated charges from the semi-empirical method AM1-BCC. AM1-BCC is the semi-empirical method where AM1 is slightly modified to reproduce the RESP charges of HF/6-31G* but at a fraction of the computational cost.

1) Change your working directory to the "charges" directory you copied in *Section A*, step 2:
   ```
   cd ~/lab3/charges/
   ```
2) We will run the semi-empirical AM1-BCC calculation with a program called **`antechamber`**. Antechamber is available as part of AmberTools, which we need to load first.
   ```
   module load cpu/0.15.4 gcc/9.2.0 openmpi/3.1.6 amber/20
   ```
   Note: the modules loaded here are different from those in **`amber_prod.q`**. This is because we intend to process the MD results on the local login node, which does not have GPUs.

3) The semiempirical calculation is very quick. We can run it directly rather than sending it through the queue to be assigned to a node with more substantial resources.
   ```
   antechamber -i h2o.pdb -fi pdb -o h2o.mol2 -fo mol2 -c bcc -s 2
   ```
   This command tells the **`antechamber`** code to read in your **`pdb`** file and generate a **`mol2`** file with AM1-BCC charges.
   Advanced tip: The Antechamber/AM1-BCC approach only works for isolated molecules, whereas the RESP server may be used to build residues that can be incorporated into a protein's backbone.
4) View the **`mol2`** file produced by **`antechamber`** with:
   ```
   nano h2o.mol2
   ```
   You should see something like this, with the location of the charges indicated in red.

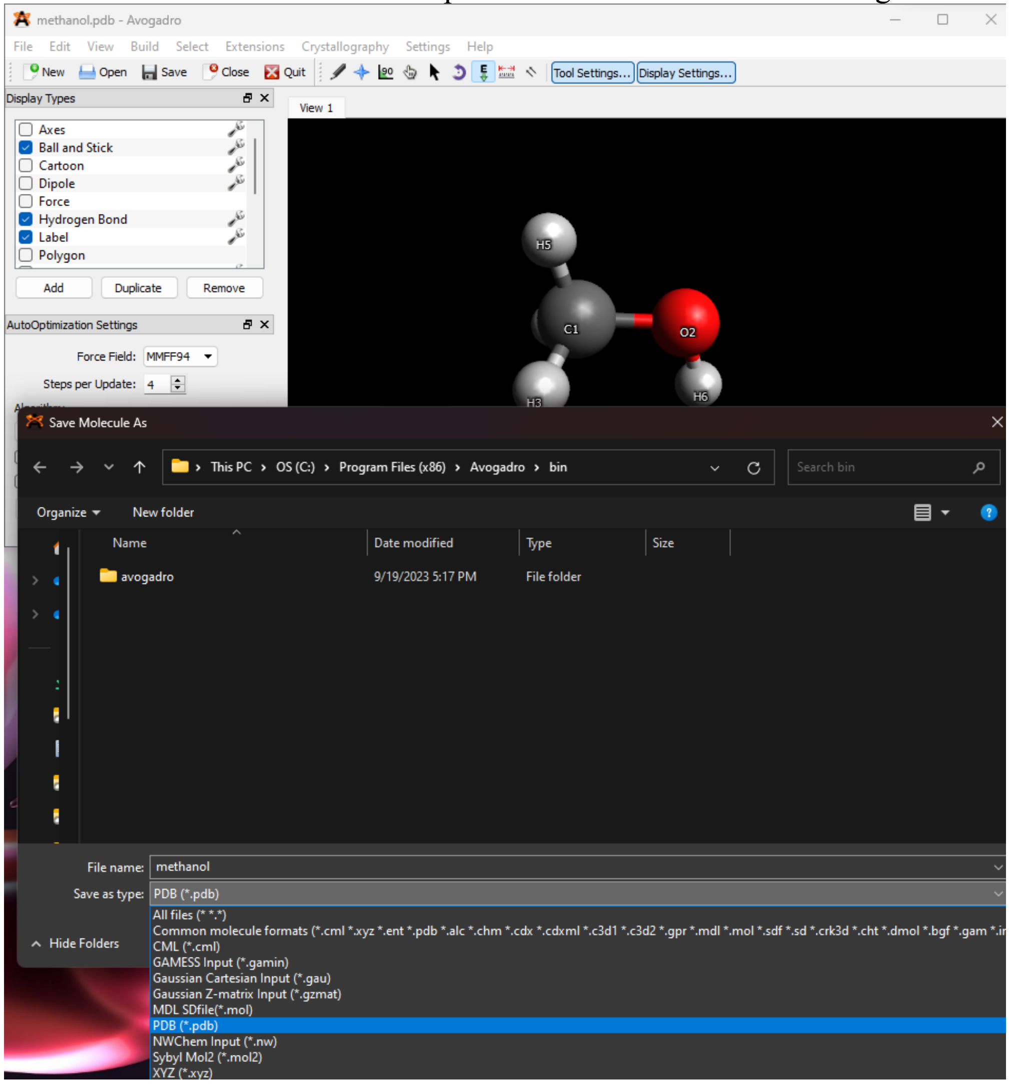
```
@<TRIPOS>MOLECULE
HOH
  3   2   1   0   0
SMALL
bcc

@<TRIPOS>ATOM
    1 O        -0.2440    1.4870    0.0200 oh        1 HOH     <oxygen charge>
    2 H         0.7240    1.4800   -0.0260 ho        0 HOH     <hydrogen charge>
    3 H1       -0.5230    1.3540   -0.8990 ho        0 HOH     <hydrogen charge>
```

```
@<TRIPOS>BOND
   1     1     2 1
   2     1     3 1
@<TRIPOS>SUBSTRUCTURE
   1 HOH        1 TEMP            0 ****  ****    0 ROOT
```
Hint: you will use these charges to answer the homework

5) For comparison, we'd like you to study the charges of methanol. First, you'll have to make a methanol molecule. In Avogadro, build and optimize methanol. Save it as **`methanol.pdb`**
   a. Use **File>Save As…**
   b. Enter **methanol** as the "File name"
   c. Choose "PDB (*.pdb)" from the drop-down menu for "Save as type"

   See *Window 1* for an example of how this could look in Avogadro.


methanol.pdb - Avogadro
File Edit View Build Select Extensions Crystallography Settings Help
New Open Save Close Quit Tool Settings... Display Settings...
Display Types
Axes
Ball and Stick
Cartoon
Dipole
Force
Hydrogen Bond
Label
Polygon
Add Duplicate Remove
AutoOptimization Settings
View 1
H5
C1
O2
H3
H6
Save Molecule As
This PC › OS (C:) › Program Files (x86) › Avogadro › bin
Search bin
Organize New folder
Name Date modified Type Size
avogadro 9/19/2023 5:17 PM File folder
Hide Folders
All files (*.*)
Common molecule formats (*.cml *.xyz *.ent *.pdb *.alc *.chm *.cdx *.cdxml *.c3d1 *.c3d2 *.gpr *.mdl *.mol *.sdf *.sd *.crk3d *.cht *.dmol *.bgf *.gam *.i
CML (*.cml)
GAMESS Input (*.gamin)
Gaussian Cartesian Input (*.gau)
Gaussian Z-matrix Input (*.gzmat)
MDL SDfile(*.mol)
PDB (*.pdb)
NWChem Input (*.nw)
Sybyl Mol2 (*.mol2)
XYZ (*.xyz)

Force Field: MMFF94
Steps per Update: 4
File name: methanol
Save as type: PDB (*.pdb)

Window 1. Saving Avogadro molecule as a **.pdb** file.

6) Transfer this file to Expanse using **`scp`**
   a. From your local machine, navigate to the folder containing the **.pdb** file.

   b. Run
```
scp methanol.pdb <ACCESS ID>@login.expanse.sdsc.edu:~/lab3/charges
```
7) Repeat steps 3 and 4 for methanol, changing the command for the antechamber calculation, for example, assuming you named your molecule **methanol.pdb**:
```
antechamber -i methanol.pdb -fi pdb -o methanol.mol2 -fo mol2 -c bcc -s 2
```

D. Processing MD Results. Now, we will analyze the results of our MD simulations of TIP3P and TIP4PFB water. For each model, we will look at the temperature, pressure, and density during the production run. We will also compare the self-diffusion coefficients and O-O radial distribution functions of each model.

1) Navigate back to your water simulations to do the analysis.
```
cd ~/lab3/water/TIP3P
```
2) To obtain temperature, pressure, and density during the production run, we first need to load AMBER locally by running th efollowing command:
```
module load cpu/0.15.4 gcc/9.2.0 openmpi/3.1.6 amber/20
```
   Then, we will be using a built-in executable provided by AMBER named `process_mdout.perl` by running
```
process_mdout.perl constP_production.out
```

   You should be able to see many files formatted as `summary.<prop>` now. As we said previously, we will only focus on temperature (`.TEMP`), pressure (`.PRES`), and density (`.DENSITY`), so only those three files are relevant.

3) To obtain the diffusion coefficient, radial distribution functions, and hydrogen bond information, we will use another built-in executable named `cpptraj`. We will start `cpptraj` with the input `prmtop` file (Note the name of the `prmtop` file is **different** for TIP3P and TIP4PFB):
   Note: You will need to refer to the "**cpptraj cheatsheet**" handout on Canvas to complete some of the analysis required in the graded lab.
```
cpptraj -p tip3pwater.prmtop
```

   By executing this command, you should be brought to a command line interface of `cpptraj`. <u>All the following commands should be run inside this `cpptraj` interface.</u> Then you need to load the trajectory file you would like to analyze,
```
trajin constpprod.mdcrd
```

   To calculate O-O radial distribution function and integrated O-O radial distribution function, we use the `radial` command. It is formatted as
```
radial <outfilename> <spacing> <maximum> <solvent mask1> [intrdf <file>]
```
   See https://amberhub.chpc.utah.edu/radial-rdf/ or the **cpptraj** handout for more information. We'd like you to specifically obtain the water RDF using the following command:
```
radial water_rdf 0.025 10 @O intrdf water_intrdf
```

   In the above, @O is a mask that tells `cpptraj` that we are interested in the oxygen atoms in the simulation.

To calculate the diffusion coefficient,

```
diffusion out diffusion time 2 * average
```

To analyze hydrogen bonds, we use the **hbond** command (all on one line),

```
hbond out hb.dat solventdonor :WAT solventacceptor :WAT@O donormask :1
  acceptormask :1@O avgout avg.dat image
```

To execute these commands, simply run,

```
go
```

This may take around 1 minute to finish. After it finishes, you should be able to quit the interface by

```
quit
```

Now you should be able to see five new files in your directory: **diffusion** for calculating the diffusion coefficient, **water_rdf** for the O-O radial distribution function, and **water_intrdf** for integrated O-O radial distribution function, **hb.dat** with the hydrogen bonding information for each time step, and **avg.dat** with the average hydrogen bonding information.

4) Finally, to obtain a more human-readable format of these files, we prepared some Python scripts (Look into them if you want to know what happens inside!) to convert them to **.csv** files. You can run them by

```
python make-csv-summary.py
python make-csv-diffusion.py
python make-csv-rdf.py
```

After you run these commands, you should be able to see 6 .csv files in total in your directory.
Note: We did not prepare a .csv file for the two hydrogen-bonding **.dat** files. Please look at these and copy them over to your local machine as needed.

5) We will copy these results files using **scp** from ACCESS to our local machines in order to take a closer look at the results. Open a terminal on your local machine and type the following commands, which will copy the results files to your current working directory:

```
scp <ACCESS ID>@login.expanse.sdsc.edu:~/lab3/water/TIP3P/*.csv ./
```

On a Mac be sure to use single quotes when using a wildcard:

```
scp <ACCESS ID>@login.expanse.sdsc.edu:'~/lab3/water/TIP3P/*.csv' ./
```

After you finish moving these files, change directories on your local machine by making a new directory before you repeat the **scp** step by replacing **TIP3P** with **TIP4PFB**. This is important to avoid overwriting files.

(Optional Windows Note) If you have WSL 2 (default for newer installations) you can view your Linux file system in the native Windows file explorer. Otherwise, you will want to copy the files over to the Windows file system which can be accessed from Linux at **/mnt/c/**

10.637: Lab3 (GL1): Classical molecular dynamics **for pure water: 30405.1626** **for Mg water: 30348.7141** **10/03/23**

very similar!

Homework assignment (GL1) for Topic 3 Lab:

Hint: You may want to refer to the "**CPPTRAJ-2023.pdf**" (available on Canvas at Module 3) file to get to know about the **cpptraj** command that you will need to use.

Boxes of water (55%)

1. (15%) Report the instantaneous and averaged pressure, temperature, and density over your production run of NPT dynamics for TIP3P and TIP4PFB water models. *Graph the instantaneous data.* Recall in class that we said that pressure was not well-defined for such small systems but that a smooth density was a better measure. Does your data agree?
2. (15%) Calculate the self-diffusion coefficient of liquid water for TIP3P and TIP4PFB. Report these in cm^2/s. Note that the units AMBER uses are ps and Å.
3. (15%) Report the O-O radial distribution function from these simulations – overlay the results from all the different water models. Comment on any differences. Determine the average number of nearest-neighbor water molecules by integrating the curve and obtaining the value at the first minimum in the curve.
4. (10%) Plot the number of hydrogen bonds between a single water molecule and the others during the NPT production run for each water model. On average, how many hydrogen bonds are formed per water molecule? What are the average lengths and angles of these bonds? Do they differ between the two water models?

Salt solution (35%)

5. (15%) Report the Mg-O radial distribution function for the TIP3P water model. Identify the number of water molecules in the first solvation shell of Mg^{2+} and the second solvation shell via the integrated RDF. Compare the Mg-O RDF to that for the Cl-O RDF. What do the differences for Mg^{2+} vs. Cl^- tell you about exchange events between first solvation shell waters and the environment in the two environments?

Hint: Think of what we talked about for the features of the RDF (or *g*(r)). Characterize your answer in terms of the nature of the peaks, peak width.

Note: the `cpptraj` comments are case-sensitive. In order to get the RDF for the O and Cl, one needs to have **"@O @Cl-".** The use of "@CL" or "@Cl" will not be recognized by `cpptraj`.

6. (5%) Assuming we instead compute the Mg-H and Cl-H RDF, how do you expect these to differ from the Mg-O and Cl-O RDFs (and from each other)? What does this imply about the orientation of water relative to the ions?

Hint: Hydrogen has a positive partial charge compared to the negative partial charge of oxygen.

7. (15%) Use `cpptraj` and the `watershell` command to identify the number of waters in each solvation shell of the Mg and Cl ions. Plot these numbers throughout the duration of the simulation. Compare this to the number identified in question 5 via the RDF and comment on the utility of the `watershell` command.

Charges (10%)

8. (10%) Compare the charges obtained from AM1-BCC to the charges you learned about for TIP3P water in class. Are they in close agreement or do they disagree? How does the AM1-BCC charge on oxygen in methanol compare to that in water? What about the hydroxyl hydrogen on methanol versus the hydrogen in water?