Tutorial

This is a short exercise designed to walk through setting up and launching an MD simulation. We will set up three different simulations, illustrating three different scenarios.

A few of notes on the format.

- Any time I refer to a *local* terminal, this means a terminal navigating your personal computer. When I refer to a *cluster* terminal, this means a terminal navigating the cluster. When you open a new terminal, it is a *local* terminal. After you type `ssh USERNAME@login.talapas.uoregon.edu`, it becomes a *cluster* terminal. (Put functionally: if you type the command `ls` on a *local* terminal, it shows files on your laptop; if you type `ls` on a *cluster* terminal, it shows files on the cluster.)
- In this document, anything in "`this font`" is text that will appear on a terminal (whether typed or printed out as output).
- Anything in "`ALL CAPS`" is a placeholder name you will likely change. For example, the instructions refer to "`PROTEIN.pdb`". You will probably have a file with a different and more descriptive name, but we will refer to it `PROTEIN.pdb` in the instructions.
- In the terminal commands, any text after a "`#`" is a comment. It is not part of a command, but can be safely pasted into a terminal because the computer will ignore any text after `#`.

Setting up

Our first task is to copy the "md-workshop" directory to talapas. Do so using `scp`:

```
scp -r md-workshop/ USERNAME@login.talapas.uoregon.edu:shared/
```

Note we copied these into the `shared` directory. **All simulations should be run within the `~/shared` directory**. The `~/shared/` folder is actually a network drive with over a petabyte of space. Your home directory, by contrast, will rapidly run out of disk space.

Next, log into the cluster.

```
ssh USERNAME@login.talapas.uoregon.edu
```

Launch an interactive job and load the `gromacs` module. This module will give us access to gromacs commands.

```
srun --account=MYPIRG --partition=compute --pty bash
module load gromacs/2023.4
```

Navigate into the directory:

```
cd ~/shared/md-workshop/
ls
```

Let's check out the script we are going to use for the calculation.

```
cd setup_md
ls
```

This should spit out the following.

```
charmm36-jul2022.ff/
run_files/
README.md
setup_md.py
```

- We are going to run `setup_md.py`. To learn more about the script, run it with the `--help` command:

  ```
  ./setup_md.py --help
  ```

  This will let you see some of the options available.

- The `charmm36-jul2022.ff` directory holds the forcefield. You can check out it's contents with `ls`:

  ```
  ls charmm36-jul2022.ff/
  ```

  This should show an incomprehensible blob of text files.

- The `run_files` directory holds the files to start our simulations.

  ```
  ls run_files
  ```

  The output should be:

  ```
  em.mdp          extract_md.srun  ions.mdp        md.mdp
     npt.mdp         nvt.mdp         run_md.srun
  ```

  The ".mdp" files tell GROMACS what to do for each simulation. The ".srun" files are used to launch jobs on the cluster.

  You need to edit the ".srun" files to use your PIRG. Open "run_md.srun" in a text editor:

  ```
  nano run_md.srun
  ```

  Alter `CHANGME` to be your PIRG (i.e. `harmslab`).

  FOR TODAY ONLY edit the `partition` to `compute` (rather than `computelong`) and the `time` to `00-04:00:00` (rather than `07-00:00:00`). This will make sure your jobs get scheduled quickly. When you do a real simulation, you should change these values back so you request a week rather than four hours of time!

```
#!/bin/bash -l
#SBATCH --account=CHANGEME      ### change this to your actual account for charging
#SBATCH --job-name=gmx_run
#SBATCH --output=hostname.out
#SBATCH --error=hostname.err
#SBATCH --partition=computelong ### change this to 'compute'
#SBATCH --time=07-00:00:00      ### change this to 00:04:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=28
```

Save the "run_md.srun" out. Then change the PIRG in "extract_md.srun". (You do not have to change the `--partition` or `--time` for that job; it is already a short job.)

Simple simulation

We will now set up a run for a protein that will work on the first try. This will let you run through the process of setting it up and allow you to see what to look for in a successful run.

Navigate to the `examples/setup` trajectory:

```
cd ~/shared/md-workshop/examples/setup
ls
```

This directory should have three directories, corresponding to the three main tasks in this tutorial.

```
01_scorpion-toxin
02_s100a4
03_sh2
tutorial.md
tutorial.pdf
```

Navigate into the `01_scorpion-toxin` directory:

```
cd 01_scorpion-toxin
ls
```

This should have a single file, `1acw.pdb`. If you open the file in a text editor, you will see it is a protein with amino acid coordinates, but no other water or solvent atoms.

```
HEADER    TOXIN                                   10-FEB-97
1ACW
TITLE     SOLUTION NMR STRUCTURE OF P01, A NATURAL SCORPION
PEPTIDE STRUCTURALLY
ATOM      1  N   VAL A   1       0.965   0.298  -0.467  1.00
1.00           N
ATOM      2  CA  VAL A   1       1.811   0.250  -1.701  1.00
1.00           C
ATOM      3  C   VAL A   1       3.290   0.400  -1.320  1.00
1.00           C
ATOM      4  O   VAL A   1       3.628   1.053  -0.346  1.00
1.00           O
ATOM      5  CB  VAL A   1       1.417   1.384  -2.664  1.00
1.00           C
ATOM      6  CG1 VAL A   1      -0.064   1.262  -3.035  1.00
1.00           C
ATOM      7  CG2 VAL A   1       1.666   2.744  -2.002  1.00
1.00           C
...
```

We will use the `setup_md.py` script to set up the simulation. The call below will set up a simulation from `1acw.pdb` and place the output into a directory called `md-input`.

```
../../../setup_md/setup_md.py md-input --standard 1acw.pdb
```

When the script runs, it will dump information to the terminal. Among the spew, look for the following:

```
Analysing hydrogen-bonding network for automated assignment of
histidine
 protonation. 40 donors and 47 acceptors were found.
There are 60 hydrogen bonds
Will use HISE for residue 9
8 out of 8 lines of specbond.dat converted successfully
Special Atom Distance matrix:
                    CYS3    CYS6    HIS9   CYS10   CYS19
CYS24
                    SG19    SG42   NE268    SG74   SG139
SG178
    CYS6    SG42   0.793
    HIS9   NE268   1.477   0.939
   CYS10    SG74   1.274   0.648   1.003
   CYS19   SG139   0.202   0.777   1.562   1.228
   CYS24   SG178   0.766   0.202   1.123   0.636   0.701
   CYS26   SG191   1.150   0.464   0.958   0.202   1.105
0.460
Linking CYS-3 SG-19 and CYS-19 SG-139...
Linking CYS-6 SG-42 and CYS-24 SG-178...
Linking CYS-10 SG-74 and CYS-26 SG-191...
```

There are two key bits of output to pay attention to when setting up a run:

- The first is the assignment of histidine protonation state. GROMACS does this by analyzing hydrogen bond networks. Each histidine in the protein will either be assigned HIS (neutral, no proton), HISD (positive one charge, protonated at N^{δ}) or HISE (positive one charge, protonated at N^{ε}). I usually use the default assignment, but it is important to know that the software is making this decision. If you need to set a different protonation state, you need to run `gmx pdb2gmx` manually rather than using this script.
- The second output is the assignment of CYS REDOX state and disulfide. When doing a run on your favorite protein, make sure these are reasonable!

The output of this script will be in the `md-input/final` directory. You can inspect it's contents with:

```
ls md-input/final
```

The following files should be there. I've broken them out and annotated them with comments (#).

```
# Assembled system coordinates
system.gro

# Forcefield parameters
charmm36-jul2022.ff
topol.top
posre.itp

# GROMACS input files for doing runs
em.mdp
nvt.mdp
npt.mdp
md.mdp
```

```
# Batch files for running jobs on cluster
run_md.srun
extract_md.srun
```

I always open `system.gro` to make sure things look reasonable before launching a simulation. Both PyMOL and VMD can read .gro files. I check that the protein is indeed in the center of a water box, that the disulfide bonds are reasonable, and make sure ligands were placed properly (in the instance we modeled in ligands).

To get the file to inspect, open a *local* terminal and run:

```
scp USERNAME@login.talapas.uoregon.edu:shared/md-
workshop/examples/setup/01_scorpion-toxin/md-
input/final/system.gro .
```

This will copy `system.gro` into whatever local directory you ran this `scp` command in. (If you're not sure what directory that is, type `pwd` to find out.) Navigate to the directory using the GUI and open `system.gro` in VMD or PyMOL. Spin it around, change the representation, etc. Make sure it looks like a protein in a box of water with ions.

Once you are satisfied that the file was generated correctly, you are ready to start an MD simulation.

Go back to your *cluster* terminal. I like to run simulations in a copy of the `final` directory so I can easily do replicate runs. Copy `md-input/final` to a directory called `run000`. (If I wanted to do another replicate, I could copy `md-input/final` to `run001`, etc. Because each simulation randomly assigns initial velocities to each atom, the outputs will be different even starting from the exact same starting structure).

```
cp -r md-input/final run000
```

You can then enter `run000` and start the run.

```
cd run000
sbatch run_md.srun
```

To see when the job starts, use `qstat`.

```
qstat -u USERNAME
```

In the `qstat` output, the `Use` column indicates the job status. `Q` means the job is queued, `R` that it's running, and `C` that it is complete. The following shows a completed job.

```
Job id               Username Queue    Name    ... Use S Time
-------------------- -------- -------- ------- ...   - -----
2758295              harms    compute  gmx_run ...   R 00:00
```

Hopefully the job start soon.

As it runs, GROMACS will start creating files corresponding to the different calculations specified in the `run_md.srun` file. The files "em.log", "nvt.log", "npt.log", and "md_0_1.log" let us track the results.

You can follow calculation progress using `tail`. The following command will follow the MD log file as it grows, allowing you to track the accumulation of simulation time.

```
tail -f md_0_1.log
```

Note that `tail -f` will go on forever. To break out of it and free up your terminal, type `CTRL-C`. (Note for macOS users: `CTRL` is really `control`, NOT the Mac `command` key).

Congratulations, you just started your first MD simulation!

Poorly named atoms

While that simulation is running, let's try setting up another run. In this one, we'll learn how to diagnose and fix a common problem that arises setting up simulations.

Navigate to the `md-workshop/examples/setup/02_s100a4/` directory on your *cluster* terminal. If you type `ls` you'll see two files: `3c1v_non-hoh.pdb` and `3c1v_no-hoh_renamed-calcium.pdb`. Run `setup_md.py` on the first pdb file:

```
../../../setup_md/setup_md.py md-input --standard 3c1v_non-
hoh.pdb
```

This will not work. You should see an error like the following:

```
-------------------------------------------------------
Program:     gmx pdb2gmx, version 2023
Source file: src/gromacs/gmxpreprocess/pdb2gmx.cpp (line 870)

Fatal error:
Atom CA in residue CA 102 was not found in rtp entry CA with 68
atoms
while sorting atoms.
.

For more information and tips for troubleshooting, please check
the GROMACS
website at http://www.gromacs.org/Documentation/Errors
-------------------------------------------------------
```

This error is telling us that GROMACS does not know how to read the file. More specifically, it's telling us that the forcefield does not have an atom named CA in a residue named CA. It calls out residue 102 as the problem. Let's dig into what's going on. Open the pdb file in your favorite text editor:

```
nano 3c1v_non-hoh.pdb
```

Scroll down to residue 102. (It's all the way at the end of the file.)

```
...
ATOM   3256  CG  PRO D  98      41.305 -46.207  17.907  1.00
32.27           C
ATOM   3257  CD  PRO D  98      42.252 -47.296  18.100  1.00
32.07           C
TER    3258      PRO D  98
HETATM 3259 CA    CA A 102      28.236 -33.634  -0.660  1.00
11.81          CA
HETATM 3260 CA    CA A 103      36.404 -30.554   6.577  1.00
10.35          CA
HETATM 3261 CA    CA B 102      34.015 -50.039  35.413  1.00
11.76          CA
HETATM 3262 CA    CA B 103      42.018 -44.326  29.743  1.00
12.00          CA
HETATM 3263 CA    CA C 102      61.382 -52.798  12.510  1.00
11.55          CA
HETATM 3264 CA    CA C 103      59.983 -44.165   5.281  1.00
10.34          CA
HETATM 3265 CA    CA D 102      45.262 -46.192 -17.899  1.00
11.67          CA
HETATM 3266 CA    CA D 103      44.295 -55.956 -23.572  1.00
11.28          CA
END
```

It turns out residue 102 is a calcium ion. GROMACS is choking when it hits it because it (apparently) does not know what a calcium ion is.

We have two options at this point. The first is to simply delete the calcium ions from the structure. Unfortunately, these calciums are critical to the biology of this molecule, so we really need to include them. Fortunately, lots of people have simulated calcium ions. These *must* be somewhere in the forcefield...

It turns out they are, but that they are not named `CA`. We can figure out what they are called in a couple of ways. The first is Google. The second is to look in the forcefield directory ourselves. Let's do that. First, navigate to the forcefield directory we visited earlier:

```
cd ../../../setup_md/charmm36-jul2022.ff/
```

We could manually open every text file looking for calcium, but that would be awful. There is a better way. The following command looks for the word "calcium" in every file, ignoring the capitalization (that's what `-i` means here).

```
grep -i calcium *
```

This reveals the answer to our problem. The calcium ion is named `CAL`, not `CA` in the CHARMM forcefield.

```
atomtypes.atp:   CAL     40.08000 ; Calcium Ion
```

If we rename those calciums to `CAL` in our .pdb file, it should work. Fortunately, I've already done this for you. Use `cd` to return to the `md-workshop/examples/setup/02_s100a4/` directory. The file `3c1v_no-hoh_renamed-calcium.pdb` has the calcium atoms renamed as `CAL`. Run the `setup_md.py` script using this file as input:

```
../../../setup_md/setup_md.py md-input --standard 3c1v_no-hoh_renamed-calcium.pdb
```

This should now work fine.

Custom parameters

In this exercise, we are going to use custom forcefield parameters to simulate something that is not a part of the CHARMM forcefield by default. Navigate to the `md-workshop/examples/setup/03_sh2` directory. The structure 1SHB.PDB is an SH2 domain bound to a phosphorylated peptide. Try to set up a run using the normal command:

```
../../../setup_md/setup_md.py md-input --standard 1shb.pdb
```

This will not work. This will print the following error.

```
Fatal error:
The residues in the chain PTR201--ALA205 do not have a
consistent type. The
first residue has type 'Other', while residue LEU202 is of type
'Protein'.
Either there is a mistake in your chain, or it includes
nonstandard residue
names that have not yet been added to the residuetypes.dat file
in the GROMACS
library directory. If there are other molecules such as
ligands, they should
not have the same chain ID as the adjacent protein chain since
it's a separate
molecule.
```

The words PTR201-205 and LEU202 hint the problem is with residues 201-205. The word "nonstandard" suggests it's a problem with the residue type. It turns out the issue is that the first residue in peptide is phosphorylated. Because this is not a standard amino acid, GROMACS does not know what parameters to use.

The solution is to generate parameters for the phosphorylated peptide. Unfortunately, showing how to do that is outside the scope of this workshop, so we are going to use some parameters I generated previously. (That said, it is relatively straightforward to generate parameters charmm-gui website. A protocol describing how to do it is in the `protocols` directory).

The `02_sh2` directory has a file called `sh2.pdb`. This is the SH2 domain by itself, with no phosphorylated peptide bound. The `pYLRVA` directory holds the phosphorylated peptide. The file `pYLRVA/model-to-pose.pdb` has the conformation and the `pYLRVA/gromacs` directory holds all of the necessary parameters. We can specify that we want to use those parameters in our simulation with the following command:

```
../../../setup_md/setup_md.py md-input --standard sh2.pdb --
custom pYLRVA/model-to-pose.pdb --params pYLRVA/gromacs/
```

This should now work and generate an `md-input` directory you can use for simulations, just like we did above for the scorpion toxin.

Extracting results

Let's go back to the scorpion toxin simulation. Navigate to `` ` ``md-workshop/examples/setup/01_scorpion-toxin/run000` directory. Hopefully by this point you have at least some output in `md_0_1.log`, indicating that the simulation has started doing unconstrained molecular dynamics. We can extract the output using the `extract_md.srun` script. Launch it with:

```
sbatch extract_md.srun
```

This script can be run co-currently with a running simulation. It will simply extract whatever simulation time has completed already. This script centers the protein inside the water box and breaks the simulation into reasonably sized chunks for downstream analysis. When it is done, it will create an `output` directory.

Once the job is done, type `ls output`. This will reveal the output:

```
traj.gro
traj.tpr
traj-split0.xtc
traj-split1.xtc
traj-split2.xtc
pre_solvate/
```

The `traj.gro` file holds our initial coordinates, `traj.tpr` holds our topology (in a compressed form), and the `traj-split*.xtc` files hold the frames of the trajectory. The number of `xtc` files will depend on the length of the simulation, as well as the `split_interval` setting in `extract_md.srun`. The "pre_solvate" directory holds the trajectory with all of the residues you specified in the initial input, but no waters or counter ions. The files are much smaller and can useful for analyses that do not rely on explicit water molecules.

Download the results to your computer. Open a *local* terminal and type:

```
scp -r
USERNAME@login.talapas.uoregon.edu:workshop/examples/setup/01_s
corpion-toxin/run000/output output
```

This will bring the results into a local `output` directory that you can then analyze on your laptop. Congrats! You've just completed your first-ever MD run!