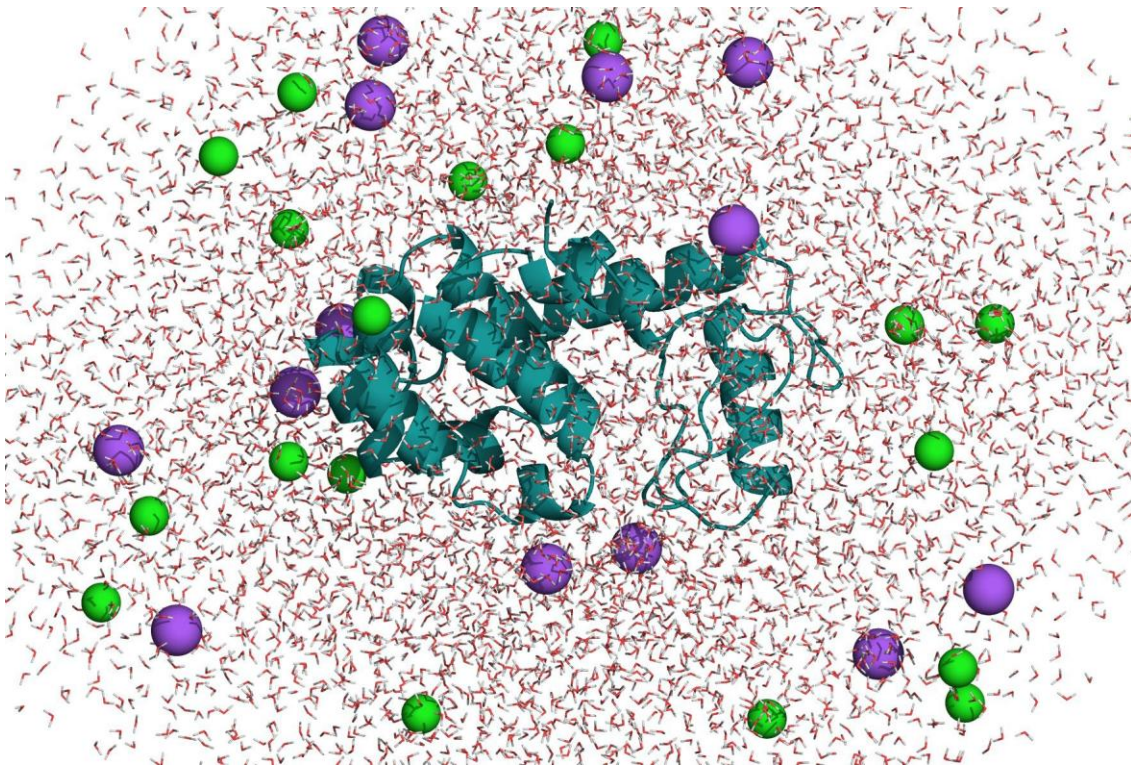


Practical 2: Molecular Dynamics simulation with GROMACS on Talapas: “Lysozyme in Water”

Part 1: Introduction to Energy Minimization



JOURNAL ARTICLE

Crystal structure of T4-lysozyme generated from synthetic coding DNA expressed in *Escherichia coli*

D.R. Rose, J. Phipps, J. Michniewicz, G.I. Birnbaum, F.R. Ahmed, A. Muir, W.F. Anderson, S. Narang

Protein Engineering, Design and Selection, Volume 2, Issue 4, October 1988, Pages 277–282, <https://doi.org/10.1093/protein/2.4.277>

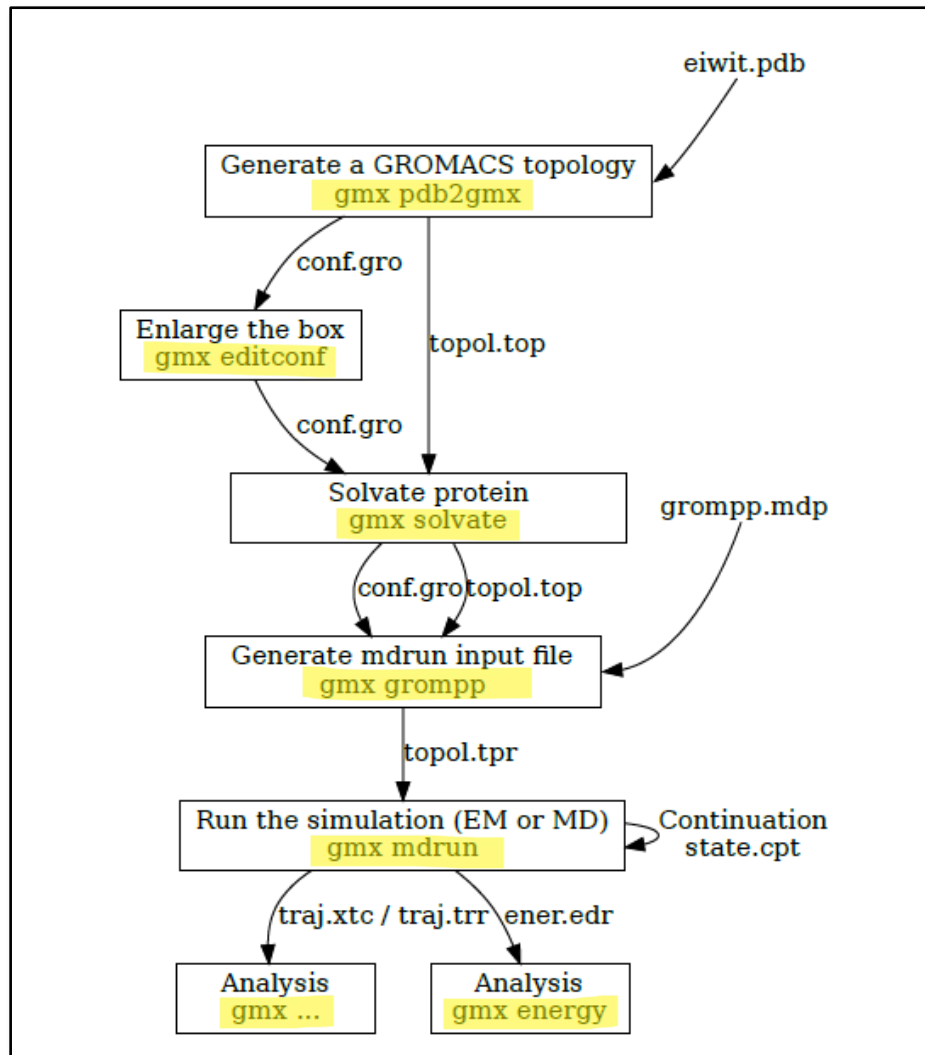
Published: 01 October 1988

Introduction

The goal of this practical is to give you an overview for the general workflow/steps used in molecular dynamics simulations. For this exercise, we can only perform short sample simulation, but you will have access to longer pre-calculated trajectory analysis(100ns).

Protein of interest for this project is the lysozyme from phage T4. This protein has served as a paradigm for studying the factors that determine the structure and stability of proteins.

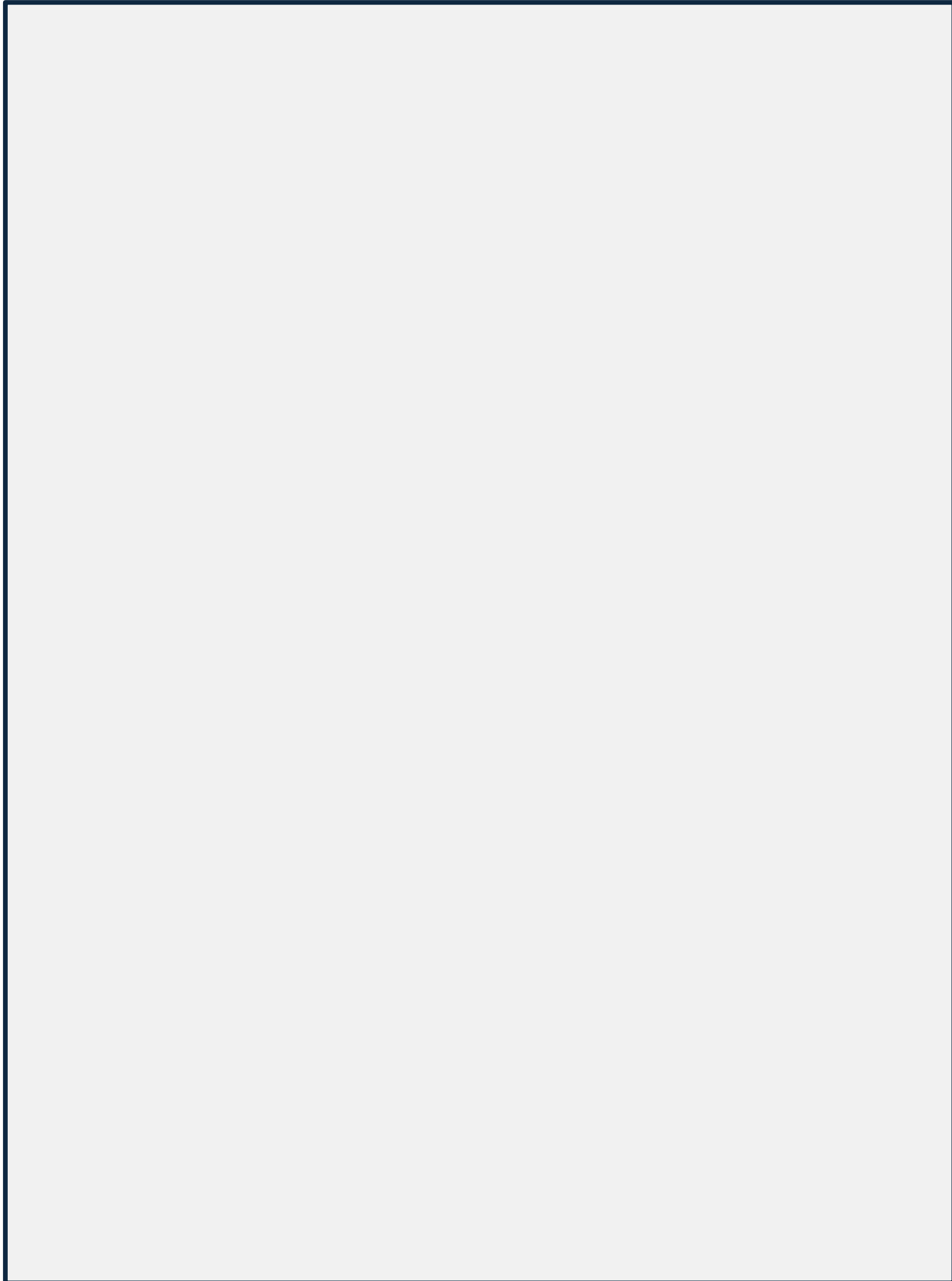
General workflow



Source: GROMACS manual

Random question: How many types of file format you see from this workflow?

“Algorithm to Simulate by”



Planning your MD simulation

Planning successful MD simulation depends on how clearly you have identified scientific question being considered or phenomena of interest to be studied by performing this simulation. Once you defined your goal, important things to consider in general is the choosing an appropriate tool(software) and force field for your system. To make good decision, you should start reading and familiarizing yourself with publications by other researchers on similar system. Also, skimming through the manual and documentations of the software to learn about the pros/cons and force field improvements, advanced analysis tools being provided, additional hardware supports such as GPU...etc. From practical point of view, you should also consider the accessibility of the software and how comfortable you are working with that tool or even learning that tool if it is new.

Preparing PDB structure for MD simulation

Define molecule of interest: DNA, RNA, protein, small ligands, polymers...etc.

In general, you can either obtain your molecule by using online databases such as Protein Data Bank (PDB) or build it by using molecular builder software: Avogadro, PyMOL, ChemDraw...etc. Different software tools require different level of preparation, and some has even built-in program that cleans up the initial input file such as Amber has LEaP,

A. Getting the PDB structure:

Go to Protein data bank(<https://www.rcsb.org/>) and search for “1LYD”, which is pdb id for T4 Lysozyme protein. Go to “Download File -> PDB Format”



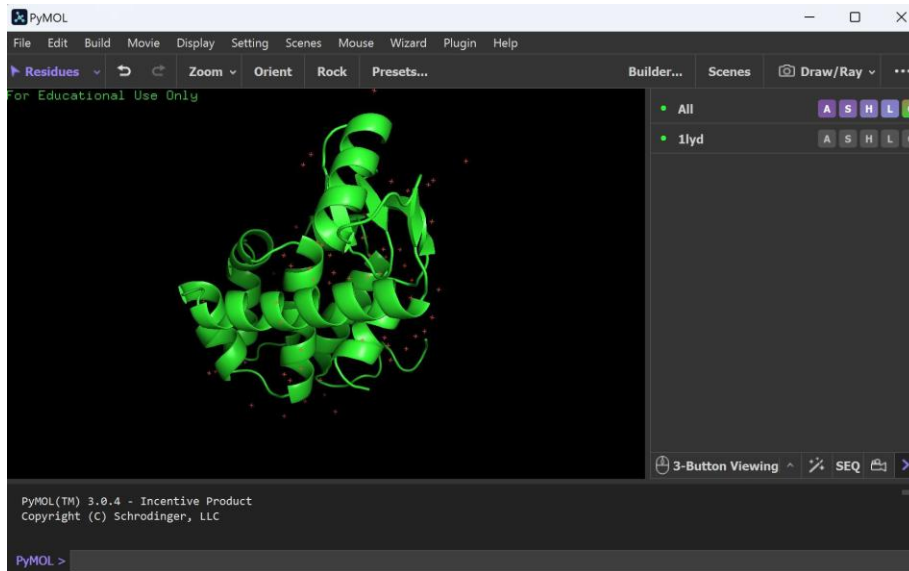
The screenshot shows the PDB entry page for 1LYD. The main title is "CRYSTAL STRUCTURE OF T4-LYSOZYME GENETICALLY MODIFIED TO BE DNA EXPRESSED IN ESCHERICHIA COLI". The classification is "HYDROLASE (O-GLYCOSYL)". The organism is "Tegulavirus T4". The deposition date is "1989-01-11" and the release date is "1990-04-15". The deposition author is "Rose, D.R.". The experimental data snapshot shows "Method: X-RAY DIFFRACTION", "Resolution: 2.00 Å", and "R-Value Observed: 0.191". The dropdown menu for "Download Files" is open, showing options like "FASTA Sequence", "PDBx/mmCIF Format", "PDB Format", and "Biological Assembly 1 (CIF - gz)".

You will see downloaded file in your computer. Open the file on PyMOL by using File-> Open on PyMOL. You will see loaded structure.

If you happen to know the PDB ID of your molecule, you can directly load it by using PyMOL command line by typing “fetch 1lyd”.

```
PyMOL> fetch 1lyd
```

In both cases, you will see loaded structure:



B. Load pdb file to Talapas folder.

You can either use scp command to transfer file from local to Talapas or you can directly transfer from Protein Data Bank by using curl command.

Open your terminal or Mobaxterm logged into Talapas:

Command: `curl https://files.rcsb.org/view/1LYD.pdb -o 1lyd.pdb`

```
[lulue@login1 TA]$ curl https://files.rcsb.org/view/1LYD.pdb -o 1lyd.pdb
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             0         0              0             0             0             0
100 140k    0 140k    0     0    703k      0 --:--:-- --:--:-- --:--:-- 703k
[lulue@login1 TA]$
```

If you downloaded 1lyd.pdb file locally and want to load that to Talapas,

Open your local terminal:

Command: `scp (local_file_path)\1lyd.pdb`

`username@login1.talapas.uoregon.edu:\Talapas_path_with_desired_file_name`

```
PS C:\Users\enhul\Desktop> cd ~
PS C:\Users\enhul> scp Desktop\1lyd.pdb lulue@login1.talapas.uoregon.edu:\gpfs\home\lulue\TA\1lyd.pdb
```

C. Just look at the file on Talapas to ensure that it is copied correctly

On Talapas, go to the directory where 1lyd.pdb file is loaded

Command: `vi 1lyd.pdb`

```
[lulue@login1 TA]$ vi 1lyd.pdb
REMARK 3
REMARK 3 ISOTROPIC THERMAL FACTOR RESTRAINTS.      RMS      SIGMA
REMARK 3 MAIN-CHAIN BOND              (A**2) : NULL ; NULL
REMARK 3 MAIN-CHAIN ANGLE              (A**2) : NULL ; NULL
REMARK 3 SIDE-CHAIN BOND              (A**2) : NULL ; NULL
REMARK 3 SIDE-CHAIN ANGLE              (A**2) : NULL ; NULL
REMARK 3
REMARK 3 OTHER REFINEMENT REMARKS: RESIDUES 162 - 164 WERE NOT VISIBLE ON
REMARK 3 THE ELECTRON DENSITY MAP. THEIR POSITIONS ARE UNCERTAIN
REMARK 4
REMARK 4 1LYD COMPLIES WITH FORMAT V. 3.30, 13-JUL-11
REMARK 100
REMARK 100 THIS ENTRY HAS BEEN PROCESSED BY BNL.
REMARK 100 THE DEPOSITION ID IS D_1000174858.
```

You can see the whole content by just clicking “Enter”. Since it is long, if you want to close the file, type **:q** (quit without saving).

Everything is good. We can prepare for simulation steps.

A. Prepare your workspace

You are probably inside `/home/ch447_547/shared/your_username` directory.

- 1.) Copy “`ch447_547/Unbiased_Simulation_Only_Codes/`” to “`ch447_547/username/`”
- 2.) Move into your new “`ch447_547/username/Unbiased_Simulation_Only_Codes/`”

This folder will contain the files...

- `1_scratch_md_generalized_positive_system.sh`
- `2_run_em_generalized.sh`
- `2_run_nvteq_generalized.sh`
- `2_run_npteq_generalized.sh`
- `1_write_GROMACS_production_codes_generalized.py`
- `base_lysozyme`

“`base_lysozyme`” folder contains codes to add a simulation box, solvate, add ions, perform energy minimization, perform short NVT equilibration run, perform short NPT equilibration run, and perform a final production run.

- 3.) Copy your pdb file of the lysozyme into the “`base_lysozyme`” folder.
- 4.) Create a “`{}_config.sh`” file which will contain information about your system. I typically name this `config.sh` file with descriptions for the specific conditions that I am running.

General Example: “`{salt_concentration}_{protein_name}_{planned_run_time}_config.sh`”

Specific Example: “`Na_100_Mg_6_lysozyme_new_forcefields_100ns_config.sh`”

In the `config.sh` file we will include the following text.

```
# config.sh

## protein related
pdb="lysozyme_1lyd.pdb"
protname="lysozyme"

## directory related
config_dir="config"
em_dir="em"
ion_dir="ions"
nvt_dir="nvt"
npt_dir="npt"
prod_dir="pro"

## salt related
na_conc=0.1
mg_conc=0.006

## run name
#run_name="Na_100_Mg_6_lysozyme_new_forcefields_100ns"
```

The file contains

- Pdb = the name of the pdb which we put in the base folder.
- Protname = Protein name which all files will be named
- Em_dir = name of energy minimization directory
- Ion_dir = name of directory for adding ions
- Nvt_dir = name of directory for running NVT simulation
- Npt_dir = name of directory for running NPT simulation
- Prod_dir = name of directory for running production run
- Na_conc = concentration of NaCl which will be added to the system
- Mg_conc = concentration of MgCl which will be added to the system

This “config.sh” file will be read by all of the following codes to produce the outputs.

B: Add Simulation Box, Waters, and Ions to Your System

5.) Run the script “1_scratch_md_generalized_positive_system.sh” by typing

```
“sbatch 1_scratch_md_generalized_positive_system.sh {base_dir} {run_name}”
```

This code writes scripts which add a simulation box, add waters, and ions to your system.

This script loads the modules to run the code

```
module purge
module load slurm
module load gromacs/2024.1
module load openmpi

module load spack-rhel8
spack load /as4cli
module load amber/22
module load python3/3.10.13
```

Amber is used to set-up the system with the latest forcefield. The python module allows you to run python codes. GROMACS is used to run the simulation.

It writes a tleap script called "tleap.in" and saves it in {run_name}/config folder

```
source leaprc.protein.ff19SB
source leaprc.water.tip3p
#source leaprc.DNA.OL21

system = loadpdb add_hydrogens.pdb
solvateBox system TIP3PBOX 14 iso
addIonsRand system Cl- 0
addIonsRand system Na+ 45 Cl- 45
addIonsRand system MG 3 Cl- 6
savePDB system ../em/lysozyme.pdb
saveAmberParm system ../em/lysozyme.parm7 ../em/lysozyme.rst7
quit
```

The tleap.in does the following steps

- a.) loads your pdb file
- b.) adds waters in a box which is 14 angstrom from each side of the protein
- c.) adds Cl- until we are at a neutral charge
- d.) adds the correct number of Na+ and Cl- ions to reach a concentration of 100mM (as specified by input)
- e.) adds the correct number of Mg(2+) and Cl- ions to reach a concentration of 6 mM (as specified by input)

The number of ions to add is calculated in the following manner...

"tleap_solvate.in" puts the protein structure in a simulation box.

The simulation box size is saved in units angstrom^3 and printed as output in "tleap_solvate.out"

For Na+

$$\# \text{ of Na}(+) \text{ ions to add} = \frac{\{\text{Na}(+) \text{ concentration in M}\} \text{ mol}}{L} \times \frac{6.022 \times 10^{-23} \text{ particles}}{\text{mol}} \times \{\text{box size}\} \text{ angstrom}^3 \times \frac{10^{-27} L}{1 \text{ angstrom}^3}$$

For Mg(2+)

$$\# \text{ of Mg}(2+) \text{ ions to add} = \frac{\{\text{Mg}(2+) \text{ concentration in M}\} \text{ mol}}{L} \times \frac{6.022 \times 10^{-23} \text{ particles}}{\text{mol}} \times \{\text{box size}\} \text{ angstrom}^3 \times \frac{10^{-27} L}{1 \text{ angstrom}^3}$$

The next steps that the script does are ...

f.) saves a pdb structure file

g.) Saves amber outputs.

Then the script converts the amber outputs to gromacs structure file (.gro).

The gromacs structure file will be used as input to run energy minimization.

- 6.) Make sure to check that no errors are given in output and that system is neutral by going into {run_name}/config/tleap.log.

You should see output that looks like...

```
8 Cl- ions required to neutralize.
Adding 8 counter ions to "system". 19924 solvent molecules will remain.
0: Placed Cl- in system at (-33.29, 34.99, 30.71).
0: Placed Cl- in system at (12.19, 13.55, -31.59).
0: Placed Cl- in system at (39.10, 36.77, -12.89).
0: Placed Cl- in system at (42.14, -24.24, 30.04).
0: Placed Cl- in system at (14.30, -26.46, 42.23).
0: Placed Cl- in system at (-9.53, -43.38, 31.38).
0: Placed Cl- in system at (-41.10, 3.53, -11.43).
0: Placed Cl- in system at (22.74, 43.36, -6.86).
```

And

```
Exiting LEaP: Errors = 0; Warnings = 2; Notes = 0.
```

C: Run Energy Minimization and NPT Simulation

- 7.) Run the script "2_run_em_generalized.sh" by typing in command

```
"sbatch 2_run_em_generalized.sh {run_name}"
```

Here is what is contained in the script

```
### runs md from scratch, copying from the base directory
module purge
module load gromacs/2024.1

run_name=$1

source "${run_name}/${run_name}_config.sh"

# energy minimization

gmx_mpi grompp -v -f ./${run_name}/${em_dir}/em.mdp \
  -c ./${run_name}/${ion_dir}/ion_${protname}.gro \
  -o ./${run_name}/${em_dir}/em_${protname}.tpr \
  -p ./${run_name}/sys.top

export OMP_NUMPY_THREADS=2

mpiexec -n 1 gmx_mpi mdrun -v -s ./${run_name}/${em_dir}/em_${protname}.tpr \
  -o ./${run_name}/${em_dir}/em_${protname}.trr \
  -c ./${run_name}/${em_dir}/em_${protname}.gro \
  -g ./${run_name}/${em_dir}/em_${protname}.log \
  -pin on -pinoffset 0

sbatch 2_run_nvteq_generalized.sh ${run_name} ${protname} ${ion_dir} ${em_dir}
```

It goes through the following steps

- a.) Loads gromacs module to run gromacs.
- b.) Sources the “_config.sh” file to use its defined variable names.
- c.) Creates a .tpr binary file which contains structure information and simulation parameter information from an em.mdp simulation parameter file and a .gro structure file (which was made in the initial set-up of the system).

The em.mdp file contains the following information

```

;
;      User spoel (236)
;      Wed Nov  3 17:12:44 1993
;      Input file
;
;define                                = -POSRE_WATER
constraints                            = none
integrator                             = steep
nsteps                                = 10000
dt = 0.002
; Output control
nstxout = 500
nstlog = 500
;      Energy Minimization
emtol                                  = 250
emstep                                 = 0.01
constraint_algorithm = shake
shake_tol                             = 0.0001
ns_type                               = grid
nstlist                               = 20
cutoff-scheme                         = Verlet
coulombtype                           = PME
rcoulomb                              = 1.0
rvdw                                  = 1.0
pbc                                   = xyz

```

It tells how many steps we will integrate for.

It tells the time-step of simulation (2 fs for this simulation).

It tells us that we will save the frames every 500 steps = 1 ps.

d.) It takes the .tpr file from the previous step and outputs a .trr trajectory file and .gro structure file.

e.) It automatically submits the next NVT simulation job to the supercomputer.

The nvt script looks like...

```

module purge
module load gromacs/2024.1

run_name=$1
protname=$2
ion_dir=$3
em_dir=$4
nvt_dir="nvt"

gmx_mpi grompp -f ./${run_name}/${nvt_dir}/nvt.mdp \
  -c ./${run_name}/${em_dir}/em_${protname}.gro \
  -r ./${run_name}/${em_dir}/em_${protname}.gro \
  -o ./${run_name}/${nvt_dir}/nvt_${protname}.tpr \
  -p ./${run_name}/sys.top -maxwarn 5

export OMP_NUM_THREADS=2

mpiexec -n 1 gmx_mpi mdrun -s ./${run_name}/${nvt_dir}/nvt_${protname}.tpr \
  -o ./${run_name}/${nvt_dir}/nvt_${protname}.trr \
  -c ./${run_name}/${nvt_dir}/nvt_${protname}.gro \
  -g ./${run_name}/${nvt_dir}/nvt_${protname}.log \
  -e ./${run_name}/${nvt_dir}/nvt_${protname}.edr \
  -pin on -pinoffset 0

sbatch 2_run_npteq_generalized.sh ${run_name} ${protname} ${ion_dir} ${em_dir}

```

It is very similar to the energy minimization script. But, it uses a nvt.mdp file as input.

The nvt.mdp file looks like...

```

;title                = NVT equilibration
define                = -DPOSRES_WATER
integrator             = md
nsteps                = 50000
dt                    = 0.002
continuation           = no

;constraint_algorithm  = lincs
constraints            = h-bonds
cutoff-scheme          = Verlet
nstlist               = 10
rcoulomb               = 1.0
rvdw                   = 1.0
coulombtype            = PME

tcoupl                 = Berendsen
tc-grps                = System
tau_t                 = 0.1
ref_t                  = 300

pcoupl                 = no

nstxout                = 500
nstvout                = 500
nstenergy              = 500
nstlog                 = 500

pbc                    = xyz
refcoord-scaling        = all

gen_vel                = yes

```

The nvt.mdp file has many more parameters than the em.mdp. An NVT simulation run is a simulation that is run with a thermostat. This simulation allows the simulation to equilibrate to reach the correct temperature as specified in the file. Here we are using Berendsen temperature coupling.

The nvt script automatically submits the next npt script.

The npt script looks like...

```
module purge
module load gromacs/2024.1

run_name=$1
protname=$2
ion_dir=$3
em_dir=$4
nvt_dir="nvt"
npt_dir="npt"

gmx_mpi grompp -f ./${run_name}/${npt_dir}/npt.mdp \
  -c ./${run_name}/${nvt_dir}/nvt_${protname}.gro \
  -r ./${run_name}/${nvt_dir}/nvt_${protname}.gro \
  -o ./${run_name}/${npt_dir}/npt_${protname}.tpr \
  -p ./${run_name}/sys.top -maxwarn 5

export OMP_NUMPY_THREADS=2

mpiexec -n 1 gmx_mpi mdrun -s ./${run_name}/${npt_dir}/npt_${protname}.tpr \
  -o ./${run_name}/${npt_dir}/npt_${protname}.trr \
  -c ./${run_name}/${npt_dir}/npt_${protname}.gro \
  -g ./${run_name}/${npt_dir}/npt_${protname}.log \
  -e ./${run_name}/${npt_dir}/npt_${protname}.edr \
  -pin on -pinoffset 0

#sbatch run_pro_GPU_gmx_mpi.sh ${run_name} ${protname} ${ion_dir} ${em_dir} ${npt_dir}
```

It is very similar to the nvt script. But, it uses a npt.mdp file as input.

The npt.mdp file looks like...

```

;title                = Lysozyme NPT equilibration
define                = -DPOSRES_WATER

integrator            = md
nsteps                = 50000
dt                    = 0.002
continuation          = yes

constraint_algorithm  = lincs
constraints            = h-bonds
cutoff-scheme         = Verlet
rcoulomb              = 1.0
rvdw                  = 1.0
coulombtype           = PME

tcoupl                = Berendsen
tc-grps               = System
tau_t                 = 0.1
ref_t                 = 300

pcoupl                = Berendsen
pcoupltype            = isotropic
tau_p                 = 2.0
ref_p                 = 1.0
compressibility       = 4.5e-5
refcoord_scaling      = all

nstxout               = 500
nstvout               = 500
nstenergy             = 500
nstlog                = 500

pbc                   = xyz

gen_vel               = no
gen_temp              = 300
gen_seed              = -1

```

The npt.mdp file has similar number of parameters as nvt.mdp. An NVT simulation run is a simulation that is run at with a thermostat and a barostat. This simulation allows the simulation to equilibrate to reach the correct temperature and pressure as specified in the file. Here we are using Berendsen temperature coupling and Berendsen pressure coupling.

8.) Make sure to check that energy has converged before going on to next step.

Output in "{run_name}/em/em_{prot_name}.log" should say something like at the end of the file.

```

Steepest Descents converged to Fmax < 250 in 9328 steps
Potential Energy    = -1.0300388e+06
Maximum force       = 2.4930473e+02 on atom 1112
Norm of force       = 4.4545775e+00

```

9.) Make sure that NPT temperature is converged and pressure is converged around 0 bar.

Navigate into {run_name}/npt folder.

Gromacs command: `gmx energy -f npt_edr.edr -o temperature.svg`

Choose temperature option

Gromacs command: `gmx energy -f npt.edr -o pressure.xvg`

Choose pressure option

You can plot them on xmgrace.

D. Run Production Run

Now we are ready to run a full length simulation. In order to run a simulation for a very long time you need to be able to submit batch jobs which start the simulation, and batch jobs that continue the simulation from where it has stopped.

We can do this using the following python command.

10.) Run the following python command.

```
"python3 1_write_GROMACS_production_codes_generalized.py -run_name {run_name} -  
prot_name {prot_name}"
```

This code does the following steps.

a.) This code creates a file in {run_name} called "run_pro_GPU_gmx_mpi_{run_name}.sh"

The file "run_pro_GPU_gmx_mpi_{run_name}.sh" looks like...

```
#!/bin/bash
#SBATCH --partition=gpu-long      ### queue to submit to
#SBATCH --job-name=run_pro_GPU_gmx_mpi_Na_100_Mg_6_lysozyme_new_forcefields_100ns      ### job name
#SBATCH --output=run_pro_GPU_gmx_mpi_Na_100_Mg_6_lysozyme_new_forcefields_100ns_jobout      ### file in which to store job stdout
#SBATCH --error=run_pro_GPU_gmx_mpi_Na_100_Mg_6_lysozyme_new_forcefields_100ns_joberr      ### file in which to store job stderr
#SBATCH --time=1-23:59:00      ### Wall clock time limit in HH:MM:SS
#SBATCH --nodes=1      ### number of nodes to use
#SBATCH --mem=32G
#SBATCH --ntasks-per-node=1      ### number of tasks to launch per node
#SBATCH --mail-user=abatela2@uoregon.edu
#SBATCH --mail-type=begin # email me when the job starts
#SBATCH --mail-type=end # email me when the job finishes
#SBATCH --account=guenzagrp      ### Account used for job submission
#SBATCH --gpus-per-task=1
#SBATCH --constraint=gpu-40gb

module purge
#module load gromacs/2024.1
module load gromacs/2024.3-plumed-2.11.0-cuda-12.4.1

run_name=$1
prot_name=$2
ion_dir=$3
em_dir=$4
npt_dir=$5
pro_dir="pro"
res_dir="res"
iteration=1

start_time=$(date +%s) # Get start time in seconds
echo "Job started at: $(date)" >> ./${pro_dir}/execution_time_pro.txt

gmx_mpi grompp -f ./${pro_dir}/pro.mdp \
-c ./${npt_dir}/npt_${prot_name}.gro \
-r ./${npt_dir}/npt_${prot_name}.gro \
-o ./${pro_dir}/pro_${prot_name}.tpr \
-p ./sys.top -maxwarn 5

export OMP_NUM_THREADS=4
#export OMP_NUM_THREADS=2

mpiexec -n 1 gmx_mpi mdrun -v -s ./${pro_dir}/pro_${prot_name}.tpr \
-g ./${pro_dir}/pro_${prot_name}.log \
-c ./${pro_dir}/pro_${prot_name}.gro \
-x ./${pro_dir}/pro_${prot_name}_traj_comp.xtc \
-e ./${pro_dir}/pro_edr.edr \
-cpo ./${pro_dir}/state.cpt \
-nsteps 50000000 \
-append \
-pin on \
-pinoffset 0 \
-gpu_id 0

#run for 50000000 steps = 100 ns
#run for 5000 steps = 0.01 ns

#the production run needs a npt/gro equilibrated topology file for -c option (fix this)

gmx dump -cp ./${pro_dir}/state.cpt > ./${pro_dir}/checkpoint_info.txt

end_time=$(date +%s) # Get end time in seconds
elapsed_time=$((end_time - start_time)) # Compute elapsed time

echo "Job ended at: $(date)" >> ./${pro_dir}/execution_time_pro.txt
echo "Elapsed time: $elapsed_time seconds" >> ./${pro_dir}/execution_time_pro.txt

next_iteration=$((iteration + 1))
sbatch run_fromrestart_Na_100_Mg_6_lysozyme_new_forcefields_100ns.sh ${run_name} ${prot_name} ${ion_dir} ${em_dir} ${npt_dir} ${pro_dir} ${next_iteration}

#Example
#sbatch run_pro_GPU_gmx_mpi.sh test_12-06-24_dialanine_ions_em_npt
#sbatch run_pro_GPU_gmx_mpi.sh Na_100_Mg_6_GP32open_1microsecond_GP32open_ions_em_npt
```

This code creates a tpr file from the pro.mdp and npt.gro file.

Then, runs the production run for 100 ns.

Saves a checkpoint file (which allows the simulation to be restarted)

Submits another sbatch job to restart the simulation where we left off.

The restart sbatch job script looks like ...

```
#!/bin/bash
#SBATCH --partition=gpu-long      ### queue to submit to
#SBATCH --job-name=run_from_restart_GPU_gmx_mpi_Na_100_Mg_6_lysozyme_new_forcefields_100ns      ### job name
#SBATCH --output=run_from_restart_GPU_gmx_mpi_Na_100_Mg_6_lysozyme_new_forcefields_100ns_jobout      ### file in which to store job stdout
#SBATCH --error=run_from_restart_GPU_gmx_mpi_Na_100_Mg_6_lysozyme_new_forcefields_100ns_joberr      ### file in which to store job stderr
#SBATCH --time=1-23:59:00      ### Wall clock time limit in HH:MM:SS
#SBATCH --nodes=1      ### number of nodes to use
#SBATCH --mem=32G
#SBATCH --ntasks-per-node=1      ### number of tasks to launch per node
#SBATCH --mail-user=abatela2@uoregon.edu
#SBATCH --mail-type=begin      # email me when the job starts
#SBATCH --mail-type=end      # email me when the job finishes
#SBATCH --account=guenzagrp      ### Account used for job submission
#SBATCH --gpus-per-task=1
#SBATCH --constraint=gpu-40gb

module purge
#module load gromacs/2024.1
module load gromacs/2024.3-plumed-2.11.0-cuda-12.4.1
#####

# other variables
run_name=$1
prot_name=$2
ion_dir=$3
em_dir=$4
npt_dir=$5
pro_dir=$6
iteration=$7

start_time=$(date +%s) # Get start time in seconds
echo "Job started at: $(date)" >> ./${pro_dir}/execution_time_pro_restart_${iteration}.txt

gmx_mpi convert-tpr -s ./${pro_dir}/pro_${prot_name}.tpr -extend 100000 -o ./${pro_dir}/pro_${prot_name}.tpr

#extend for 100000; means extend 100000 ps = 100 ns
#extend for 10; means means extend 10 ps

#export OMP_NUM_THREADS=2
export OMP_NUM_THREADS=4

mpirun -n 1 gmx_mpi mdrun -v -s ./${pro_dir}/pro_${prot_name}.tpr \
-cpi ./${pro_dir}/state.cpt \
-g ./${pro_dir}/pro_${prot_name}.log \
-x ./${pro_dir}/pro_${prot_name}_traj_comp.xtc \
-e ./${pro_dir}/pro_edr.edr \
-cpo ./${pro_dir}/state.cpt \
-append \
-pln on \
-plnoffset 0 \
-gpu_id 0

gmx dump -cp ./${pro_dir}/state.cpt > ./${pro_dir}/checkpoint_info.txt

end_time=$(date +%s) # Get end time in seconds
elapsed_time=$((end_time - start_time)) # Compute elapsed time

echo "Job ended at: $(date)" >> ./${pro_dir}/execution_time_pro_restart_${iteration}.txt
echo "Elapsed time: $elapsed_time seconds" >> ./${pro_dir}/execution_time_pro_restart_${iteration}.txt

if [ $iteration -lt 3 ]
then
    next_iteration=$((iteration + 1))
    sbatch run_fromrestart_Na_100_Mg_6_lysozyme_new_forcefields_100ns.sh $run_name $prot_name $ion_dir $em_dir $npt_dir $pro_dir $next_iteration
    exit
else
    echo 'jobs done'
    exit
fi

#Example
#sbatch run_fromrestart.sh test_12-06-24 dialanine ions em npt
#sbatch run_fromrestart.sh Na_100_Mg_6_GP32open_1microsecond GP32open ions em npt pro 2
#sbatch run_fromrestart.sh Na_100_Mg_6_GP32closed_1microsecond GP32closed ions em npt pro 3
```

This script

Extends the tpr file by 100ns

Restarts the simulation from where we left off

Saves a checkpoint file (which allows restarts)

Submits a batch script to restart the simulation again (if iteration number is less than 3)

Summary of Instructions

- 1.) Copy "ch447_547/Unbiased_Simulation_Only_Codes/" to "ch447_547/username/"
- 2.) Move into your new "ch447_547/username/Unbiased_Simulation_Only_Codes/"
- 3.) Copy your pdb file of the lysozyme into the "base_lysozyme" folder.
- 4.) Create a "{}_config.sh" file which will contain information about your system. I typically name this config.sh file with descriptions for the specific conditions that I am running.

General Example: "{salt_concentration}_{protein_name}_{planned_run_time}_config.sh

Specific Example: "Na_100_Mg_6_lysozyme_new_forcefields_100ns_config.sh"

In the config.sh file we will include the following text.

```
# config.sh

## protein related
pdb="lysozyme_1lyd.pdb"
protname="lysozyme"

## directory related
config_dir="config"
em_dir="em"
ion_dir="ions"
nvt_dir="nvt"
npt_dir="npt"
prod_dir="pro"

## salt related
na_conc=0.1
mg_conc=0.006

## run name
#run_name="Na_100_Mg_6_lysozyme_new_forcefields_100ns"
```

- 5.) Run the script "1_scratch_md_generalized_positive_system.sh" by typing
"sbatch 1_scratch_md_generalized_positive_system.sh {base_dir} {run_name}"
- 6.) Make sure to check that no errors are given in output and that system is neutral by going into {run_name}/config/tleap.log.
- 7.) Run the script "2_run_em_generalized.sh" by typing in command
"sbatch 2_run_em_generalized.sh {run_name}"
- 8.) Make sure to check that energy has converged before going on to next step.
- 9.) Check if temperature has stabilized and pressure has stabilized around 0 bar at end of npt simulation.
- 10.) Run the following python command.
"python3 1_write_GROMACS_production_codes_generalized.py -run_name {run_name} -prot_name {prot_name}"