Anirudh Rao
BE21B004

BT5420

Computer Simulations of Biomolecular Systems

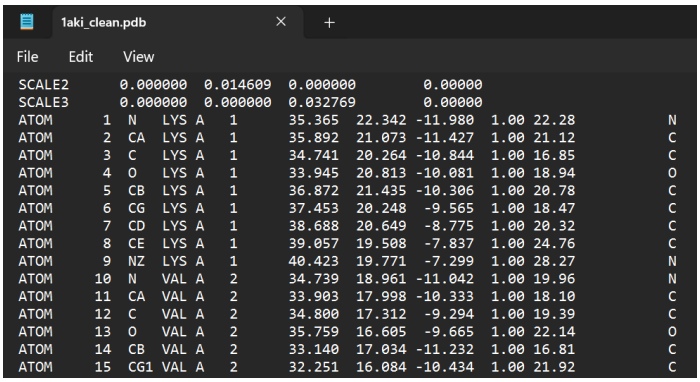
Assignment 2 (GROMACS)

1. **`pdb2gmx`** is used to add hydrogens to the molecule given as input. This can be used to build missing hydrogen coordinates in the input coordinate file (`.pdb`). Already existing hydrogen atoms that may be present in the input coordinate file can be ignored using the **`-ignh` flag**. Thus, `pdb2gmx` ignores the already existing hydrogen atoms in the input file, automatically adds the necessary hydrogen coordinates, and gives an output `.gro` file. It is assumed that the input `.pdb` file has had its crystal waters removed earlier (by running the command `grep -v HOH 1aki.pdb > 1aki_clean.pdb`). The syntax to run `pdb2gmx` is

```
gmx pdb2gmx -f 1aki_clean.pdb -o 1aki_processed.gro -water
spce -ignh
```

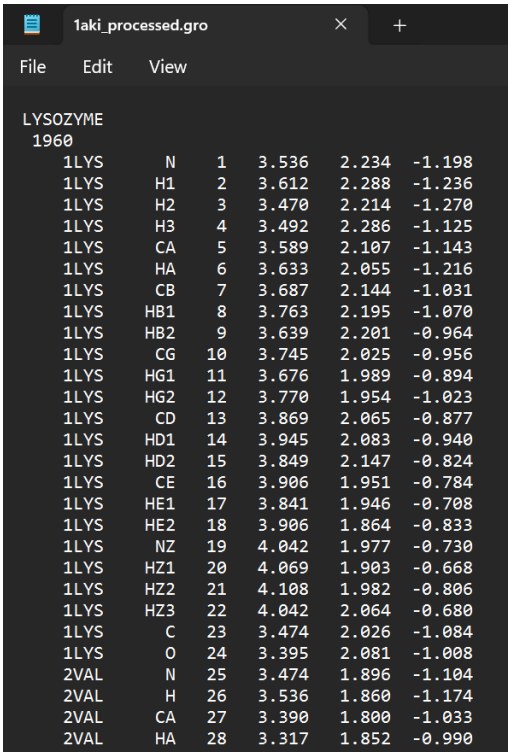
   The flag `-f` is used to denote the input file and `-o` is used to denote the output file. `-water` is used to set the water model. After running this, we will be prompted to choose a force field. We will pick the all-atom OPLS force field, by typing and entering the number 15.

2. The `.gro` file produced by `pdb2gmx` is much more compact than the input `.pdb` file. All the remarks given in the `.pdb` file are erased and only the atomic coordinates are retained. The `.pdb` file may or may not have hydrogen coordinates but the `.gro` file includes hydrogen coordinates. The `.gro` file complies with a user-defined force field. The coordinates in `.pdb` are in **angstrom** units while those in the `.gro` file are in **nanometre** units.



```
SCALE2       0.000000  0.014609  0.000000        0.00000
SCALE3       0.000000  0.000000  0.032769        0.00000
ATOM      1  N   LYS A   1       35.365  22.342 -11.980  1.00 22.28           N
ATOM      2  CA  LYS A   1       35.892  21.073 -11.427  1.00 21.12           C
ATOM      3  C   LYS A   1       34.741  20.264 -10.844  1.00 16.85           C
ATOM      4  O   LYS A   1       33.945  20.813 -10.081  1.00 18.94           O
ATOM      5  CB  LYS A   1       36.872  21.435 -10.306  1.00 20.78           C
ATOM      6  CG  LYS A   1       37.453  20.248  -9.565  1.00 18.47           C
ATOM      7  CD  LYS A   1       38.688  20.649  -8.775  1.00 20.32           C
ATOM      8  CE  LYS A   1       39.057  19.508  -7.837  1.00 24.76           C
ATOM      9  NZ  LYS A   1       40.423  19.771  -7.299  1.00 28.27           N
ATOM     10  N   VAL A   2       34.739  18.961 -11.042  1.00 19.96           N
ATOM     11  CA  VAL A   2       33.903  17.998 -10.333  1.00 18.10           C
ATOM     12  C   VAL A   2       34.800  17.312  -9.294  1.00 19.39           C
ATOM     13  O   VAL A   2       35.759  16.605  -9.665  1.00 22.14           O
ATOM     14  CB  VAL A   2       33.140  17.034 -11.232  1.00 16.81           C
ATOM     15  CG1 VAL A   2       32.251  16.084 -10.434  1.00 21.92           C
```

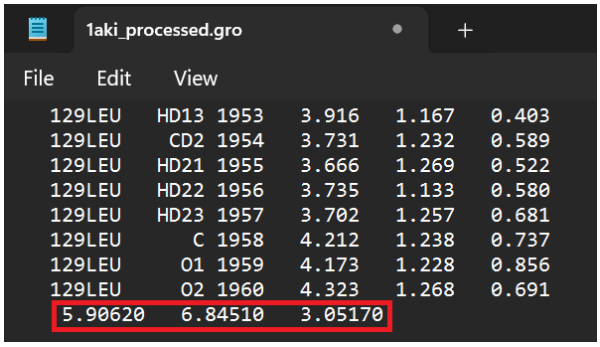
Figure 1: `.pdb` file for 1AKI



```
LYSOZYME
 1960
    1LYS      N    1   3.536   2.234  -1.198
    1LYS     H1    2   3.612   2.288  -1.236
    1LYS     H2    3   3.470   2.214  -1.270
    1LYS     H3    4   3.492   2.286  -1.125
    1LYS     CA    5   3.589   2.107  -1.143
    1LYS     HA    6   3.633   2.055  -1.216
    1LYS     CB    7   3.687   2.144  -1.031
    1LYS    HB1    8   3.763   2.195  -1.070
    1LYS    HB2    9   3.639   2.201  -0.964
    1LYS     CG   10   3.745   2.025  -0.956
    1LYS    HG1   11   3.676   1.989  -0.894
    1LYS    HG2   12   3.770   1.954  -1.023
    1LYS     CD   13   3.869   2.065  -0.877
    1LYS    HD1   14   3.945   2.083  -0.940
    1LYS    HD2   15   3.849   2.147  -0.824
    1LYS     CE   16   3.906   1.951  -0.784
    1LYS    HE1   17   3.841   1.946  -0.708
    1LYS    HE2   18   3.906   1.864  -0.833
    1LYS     NZ   19   4.042   1.977  -0.730
    1LYS    HZ1   20   4.069   1.903  -0.668
    1LYS    HZ2   21   4.108   1.982  -0.806
    1LYS    HZ3   22   4.042   2.064  -0.680
    1LYS      C   23   3.474   2.026  -1.084
    1LYS      O   24   3.395   2.081  -1.008
    2VAL      N   25   3.474   1.896  -1.104
    2VAL      H   26   3.536   1.860  -1.174
    2VAL     CA   27   3.390   1.800  -1.033
    2VAL     HA   28   3.317   1.852  -0.990
```

Figure 2: The corresponding `.gro` file for 1AKI after running `pdb2gmx`

3. The current box dimensions, i.e., before solvation, are shown below:



```
  129LEU   HD13 1953   3.916   1.167   0.403
  129LEU    CD2 1954   3.731   1.232   0.589
  129LEU   HD21 1955   3.666   1.269   0.522
  129LEU   HD22 1956   3.735   1.133   0.580
  129LEU   HD23 1957   3.702   1.257   0.681
  129LEU      C 1958   4.212   1.238   0.737
  129LEU     O1 1959   4.173   1.228   0.856
  129LEU     O2 1960   4.323   1.268   0.691
   5.90620   6.84510   3.05170
```

Figure 3: The box dimensions before solvation

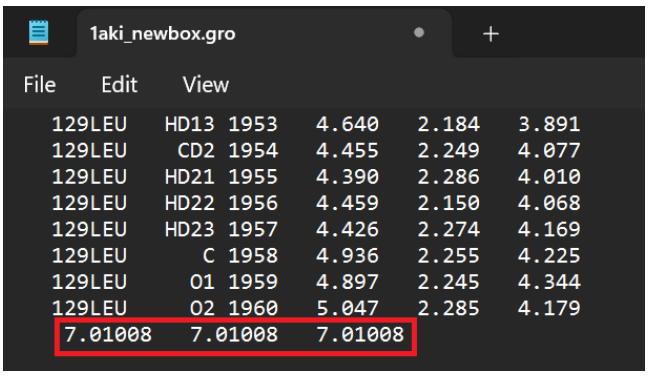
Before solvating the system, we have to first define a box with appropriate geometry and padding. This can be done by running the command

```
gmx editconf -f 1aki_processed.gro -o 1aki_newbox.gro -c -d
1.0 -bt cubic
```

`-f` and `-o` specify the input and output `.gro` files respectively. `-c` centers the protein in the box. `-d` specifies the padding distance, i.e., the minimum distance (in nanometres) between the protein and the box edge. A padding

of 1.0 nm means that there is at least a 2.0 nm distance between any two periodic images of the protein. `-bt` specifies the box type, which in this case is `cubic`.

After running this command, the box dimensions change to:



```
1aki_newbox.gro
File   Edit   View
  129LEU   HD13 1953   4.640   2.184   3.891
  129LEU    CD2 1954   4.455   2.249   4.077
  129LEU   HD21 1955   4.390   2.286   4.010
  129LEU   HD22 1956   4.459   2.150   4.068
  129LEU   HD23 1957   4.426   2.274   4.169
  129LEU      C 1958   4.936   2.255   4.225
  129LEU     O1 1959   4.897   2.245   4.344
  129LEU     O2 1960   5.047   2.285   4.179
  7.01008   7.01008   7.01008
```

Figure 4: The box dimensions after running `editconf`

Thus, the box dimensions needed to solvate the system is **7.01008, 7.01008, 7.01008**, along the x, y and z directions respectively.

We can now solvate the system by running the command:

```
gmx solvate -cp 1aki_newbox.gro -cs spc216.gro -o
            1aki_solv.gro -p topol.top
```

`-cp` specifies the conformation of the protein in the box, which is contained in the output of the previous `editconf` step.

`-cs` specifies the conformation of the solvent, which is inbuilt into GROMACS. In this case, we are using the spc216 water solvent (SPC = simple point charge).

`-o` specifies the output `.gro` file which will contain the solvated protein.

`-p` specifies the topology file (which is generated by `pdb2gmx`) that will have to be modified after solvation.

We can visualise the solvated box with the protein in VMD. To do so, we have to first load the `1aki_solv.gro` file in VMD. We can represent the protein with NewCartoon and the solvent with Lines as the Drawing Method. Then, in the Tk Console , we can run the command

```
draw pbcbox
```

This will allow us to visualise the solvated protein in the box.

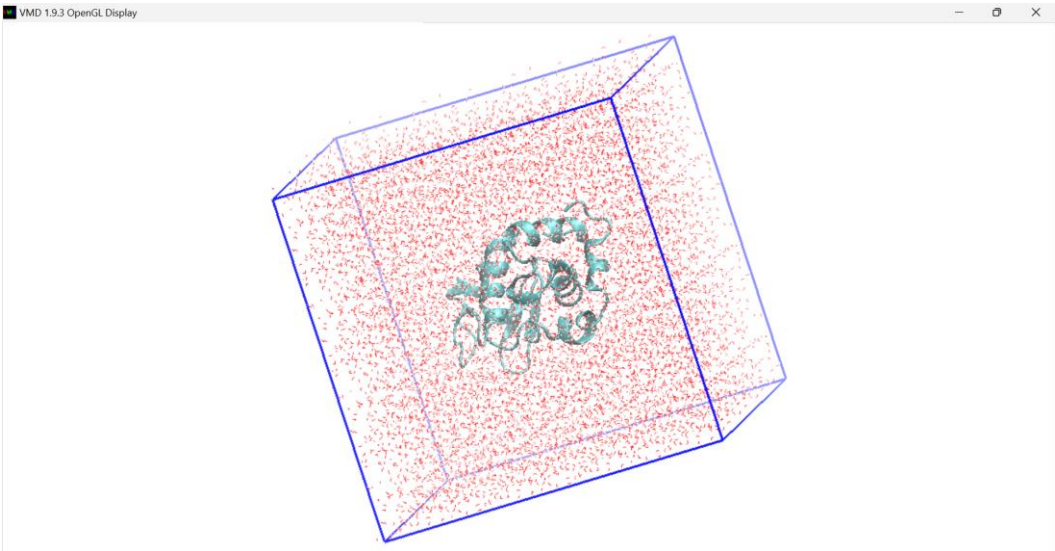


Figure 5: The solvated protein in the cubic box

4. Currently, the protein has a net charge of +8 according to the `topol.top` file. We have to neutralise this charge. To do so, we have to first generate a binary input file which has the extension `.tpr`. Using `grompp`, we will process the coordinate `.gro` file and topology `.top` file to generate an atomic-level `.tpr` input file. This requires an additional `.mdp` file which contains the simulation parameters to be used while neutralising the system. To generate the `.tpr` file, we have to run the command

```
gmx grompp -f ions.mdp -c 1aki_solv.gro -p topol.top -o
ions.tpr
```

`-f` specifies the `.mdp` file to be used as MD parameters.

`-c` specifies the coordinate file (`.gro`) of the solvated protein.

`-p` specifies the topology `.top` file.

`-o` specifies the output `.tpr` file.

Next, we take this binary `.tpr` file and neutralise the protein by running the command

```
gmx genion -s ions.tpr -o 1aki_solv_ions.gro -p topol.top -
pname NA -nname CL -neutral
```

`-s` specifies the `.tpr` file that contains the current state of the protein in terms of atomic-level descriptions.

`-o` specifies the output `.gro` file.

`-p` specifies the topology `.top` file to modify.

`-pname` specifies the positive ion to be used for neutralisation. In this case, Na^+.

`-nname` specifies the negative ion to be used for neutralisation. In this case, Cl^-.

`-neutral` specifies that we must add enough ions to neutralise the system.

When the command is run, we will be prompted to select a group of existing atoms in the current structure from which member atoms will be replaced with the ions. Since we are replacing the solvent with ions, we choose Group 13 – `SOL`.

Since the protein had a charge of +8, `genion` will introduce 8 negative ions, i.e., chloride ions, in order to neutralise the system. This has the **resname identifier `CL`**.

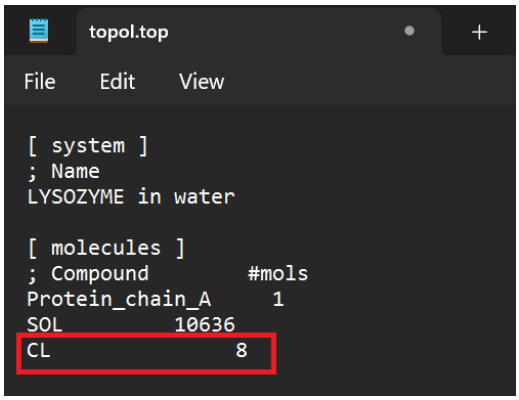


Figure 6: 8 `CL` ions added in the `topol.top` file after running `genion`

5. The solvated and neutral protein needs to have its structure relaxed by minimising its energy. This ensures that there are no steric clashes or incorrect geometries. First, we use `grompp` to generate the `.tpr` file required for the energy minimisation. This uses the parameters in the `minim.mdp` file provided. We have to run the command

```
gmx grompp -f minim.mdp -c 1aki_solv_ions.gro -p topol.top -o
em.tpr
```

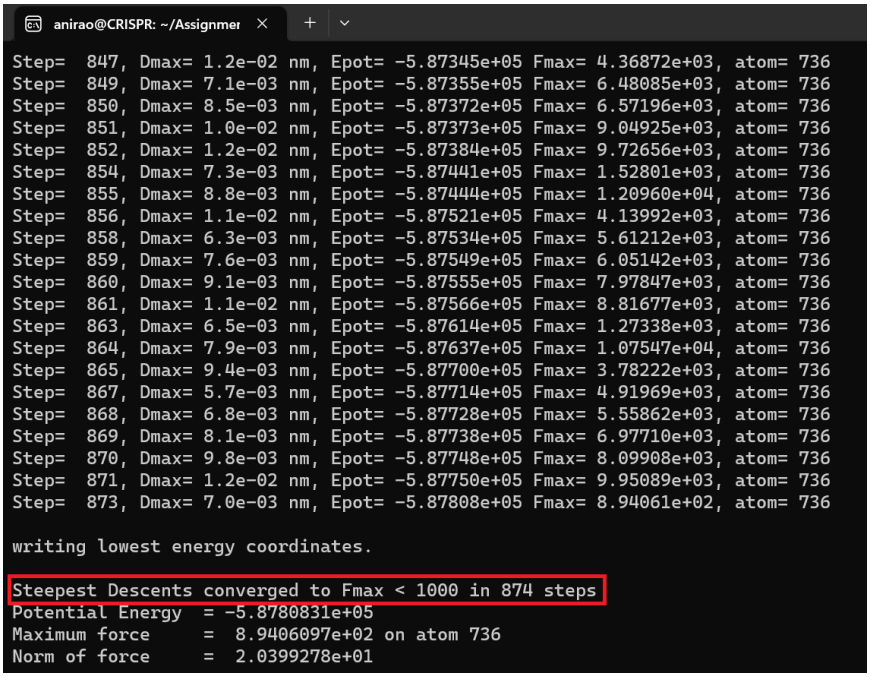
Next, we can minimise the energy by running the command

```
gmx mdrun -v -deffnm em
```

`-v` makes `mdrun` verbose, such that it prints its progress to the screen at every step.

`-deffnm` defines the default file names of the input and output. Since the output of the previous `grompp` step is `em.tpr`, we mention the file name as `em`.

Although `minim.mdp` specifies the number of steps as 50000, the energy minimisation process **converges in 874 steps** as the potential energy does not change much after around 800 steps.



```
anirao@CRISPR: ~/Assignmer
Step=  847, Dmax= 1.2e-02 nm, Epot= -5.87345e+05 Fmax= 4.36872e+03, atom= 736
Step=  849, Dmax= 7.1e-03 nm, Epot= -5.87355e+05 Fmax= 6.48085e+03, atom= 736
Step=  850, Dmax= 8.5e-03 nm, Epot= -5.87372e+05 Fmax= 6.57196e+03, atom= 736
Step=  851, Dmax= 1.0e-02 nm, Epot= -5.87373e+05 Fmax= 9.04925e+03, atom= 736
Step=  852, Dmax= 1.2e-02 nm, Epot= -5.87384e+05 Fmax= 9.72656e+03, atom= 736
Step=  854, Dmax= 7.3e-03 nm, Epot= -5.87441e+05 Fmax= 1.52801e+03, atom= 736
Step=  855, Dmax= 8.8e-03 nm, Epot= -5.87444e+05 Fmax= 1.20960e+04, atom= 736
Step=  856, Dmax= 1.1e-02 nm, Epot= -5.87521e+05 Fmax= 4.13992e+03, atom= 736
Step=  858, Dmax= 6.3e-03 nm, Epot= -5.87534e+05 Fmax= 5.61212e+03, atom= 736
Step=  859, Dmax= 7.6e-03 nm, Epot= -5.87549e+05 Fmax= 6.05142e+03, atom= 736
Step=  860, Dmax= 9.1e-03 nm, Epot= -5.87555e+05 Fmax= 7.97847e+03, atom= 736
Step=  861, Dmax= 1.1e-02 nm, Epot= -5.87566e+05 Fmax= 8.81677e+03, atom= 736
Step=  863, Dmax= 6.5e-03 nm, Epot= -5.87614e+05 Fmax= 1.27338e+03, atom= 736
Step=  864, Dmax= 7.9e-03 nm, Epot= -5.87637e+05 Fmax= 1.07547e+04, atom= 736
Step=  865, Dmax= 9.4e-03 nm, Epot= -5.87700e+05 Fmax= 3.78222e+03, atom= 736
Step=  867, Dmax= 5.7e-03 nm, Epot= -5.87714e+05 Fmax= 4.91969e+03, atom= 736
Step=  868, Dmax= 6.8e-03 nm, Epot= -5.87728e+05 Fmax= 5.55862e+03, atom= 736
Step=  869, Dmax= 8.1e-03 nm, Epot= -5.87738e+05 Fmax= 6.97710e+03, atom= 736
Step=  870, Dmax= 9.8e-03 nm, Epot= -5.87748e+05 Fmax= 8.09908e+03, atom= 736
Step=  871, Dmax= 1.2e-02 nm, Epot= -5.87750e+05 Fmax= 9.95089e+03, atom= 736
Step=  873, Dmax= 7.0e-03 nm, Epot= -5.87808e+05 Fmax= 8.94061e+02, atom= 736

writing lowest energy coordinates.

Steepest Descents converged to Fmax < 1000 in 874 steps
Potential Energy  = -5.8780831e+05
Maximum force     =  8.9406097e+02 on atom 736
Norm of force     =  2.0399278e+01
```

Figure 7: Number of steps required for energy minimisation after running `mdrun`

To analyse the change in potential energy during minimisation, we can run the command

```
gmx energy -f em.edr -o potential.xvg
```

This command takes in the binary energy file `em.edr` generated by `mdrun` as input and returns a `.xvg` file containing the potential energy at each step. When this command is run, we will be prompted to choose the terms

we wish to analyse. We have to select 10 and hit Enter to analyse the potential energy.

The contents of `potential.xvg` can be plotted using Python.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

x,y = np.loadtxt("potential.xvg",comments=["@","#"],unpack=True)

plt.figure(dpi=100)
plt.scatter(x,y)
plt.xlabel("Steps")
plt.ylabel("Potential Energy");
```

Figure 8: Python code to plot potential energy

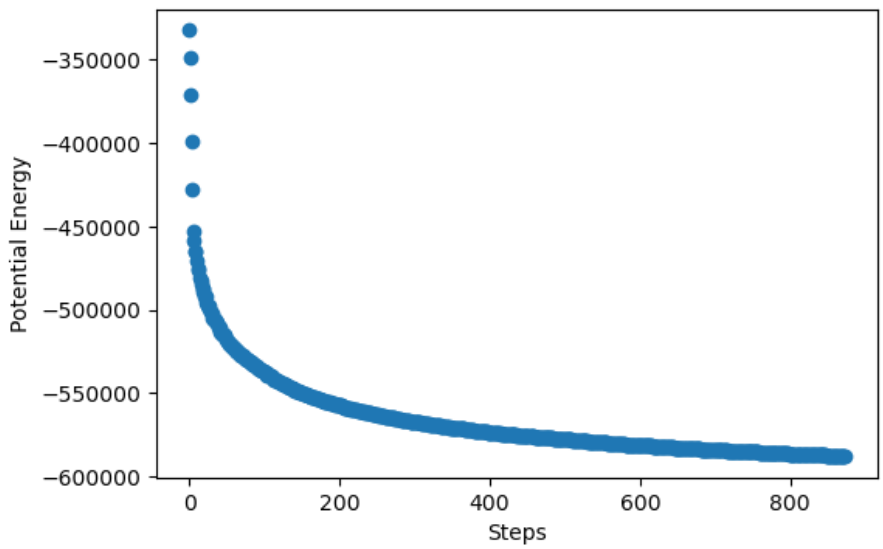


Figure 9: Potential energy as a function of step number during energy minimisation

The input `1aki_solv_ions.gro` file and the output `em.gro` file can be compared using VMD. We can load them and represent them with different colours using ColorID as the Colouring Method. We represent both of them with NewCartoon as the Drawing Method.

We represent the input structure in blue and the energy minimised structure in red. We can see that there are **no major changes in the protein's structure**.

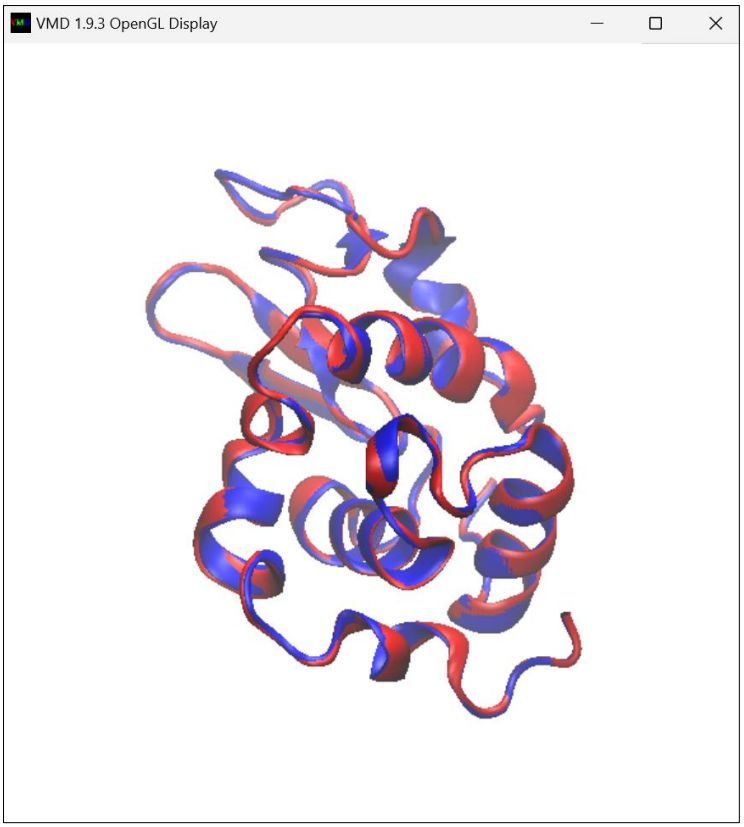


Figure 10: Structural differences between the input structure (blue) and energy minimised structure (red)

6. In case a simulation suddenly stops, the user can still restore the progress from the checkpoint (`.cpt`) file. The checkpoint files contain a complete description of the system, i.e., the precise coordinates and velocities of the system. By default, GROMACS creates a checkpoint file of the system every 15 minutes. Therefore, we can always continue the simulation from the last checkpoint by specifying the `.cpt` file with the `-cpi` flag of `mdrun`.

```
gmx mdrun -v -deffnm <file_name> -cpi <file_name>.cpt
```

7. A padding of 1.0 nm during the solvation step is necessary to ensure that the protein in the box has no interaction with the protein in the adjacent unit cells. If such interactions occur, all energy calculations will be incorrect. Hence, creating a padding of 1.0 nm ensures that any two images of the protein are at least 2.0 nm apart from each other, which is sufficient for a molecular dynamics simulation.

   If the padding is 0.5 nm, there will be unwanted protein-protein self-interactions. If the padding is 2.0 nm, the box size will be too large and the simulation can become computationally expensive.

8. PBC stands for periodic boundary conditions. The system to be simulated is placed in a box called a unit cell, which is surrounded by translated copies of itself. This enables us to simulate an infinite system by treating a part of the system as a periodically repeating unit. During the simulation, atoms are free to move in the unit cell, and their periodic images in the adjacent cells move in an identical way. This means that when any atom crosses a boundary of the cell, it will reappear on the opposite side.

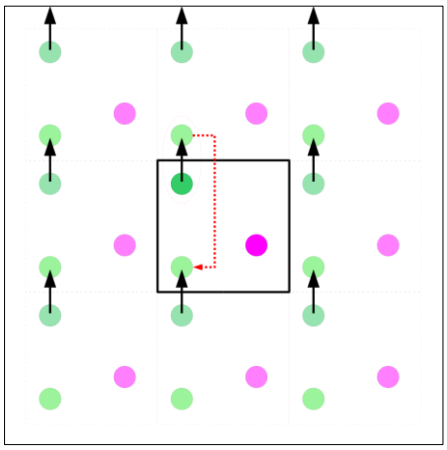


Figure 11: Periodic boundary conditions in 2D (Source: Wikimedia Commons)

Before analysing the trajectory of a molecular dynamics simulation, it is important to remove the box created for PBC. During the simulation, some of the atoms in the protein may move out of the box. Performing analysis only for those atoms that remain in the box will give rise to erroneous results as the protein will appear to be broken. Hence, we remove the box and only follow the continuous trajectory of the atoms that were originally in the box. This can be achieved by running the command

```
gmx trjconv -s md_0_1.tpr -f md_0_1.xtc -o md_0_1_noPBC.xtc -
                    pbc mol -center
```

`-s` specifies the structure file.

`-f` specifies the input trajectory file, which has the PBC.

`-o` specifies the output trajectory file, with removed PBC.

`-pbc` specifies the periodic boundary conditions used. In this case, `mol` refers to the fact that the centre of mass of all the molecules is within the box.

`-center` centers the atoms in the box.

On being prompted, we have to select 1 ("Protein") as the group to be centered and 0 ("System") for output.

9. To run NVT equilibration, we first generate the binary `.tpr` file using `grompp`. The command to be run is

```
gmx grompp -f nvt.mdp -c em.gro -r em.gro -p topol.top -o nvt.tpr
```

`-r` specifies the structure file with restrained coordinates.

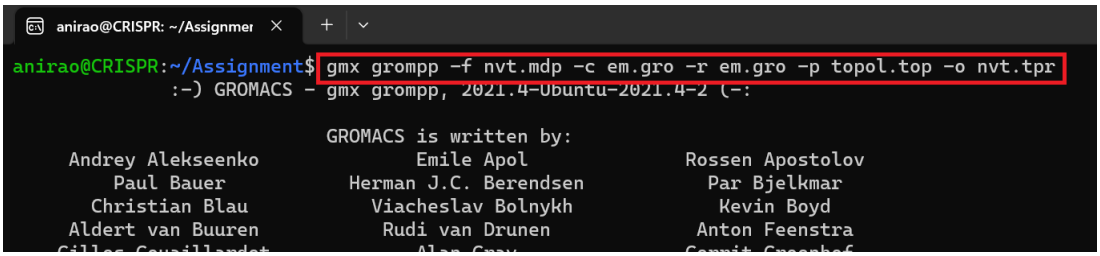


Figure 12: Command to generate `nvt.tpr` by running `grompp`

Next, we have to run the simulation using `mdrun`.

```
gmx mdrun -v -deffnm nvt
```

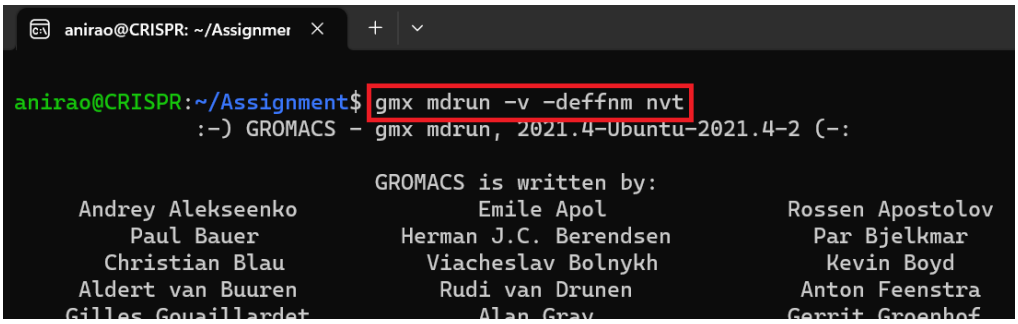


Figure 13: Command to run NVT equilibration using `mdrun`

After the equilibration is completed, we can find the average temperature by running the command

```
gmx energy -f nvt.edr -o temperature.xvg
```

When prompted, we have to select 16 and hit Enter to analyse the temperature.

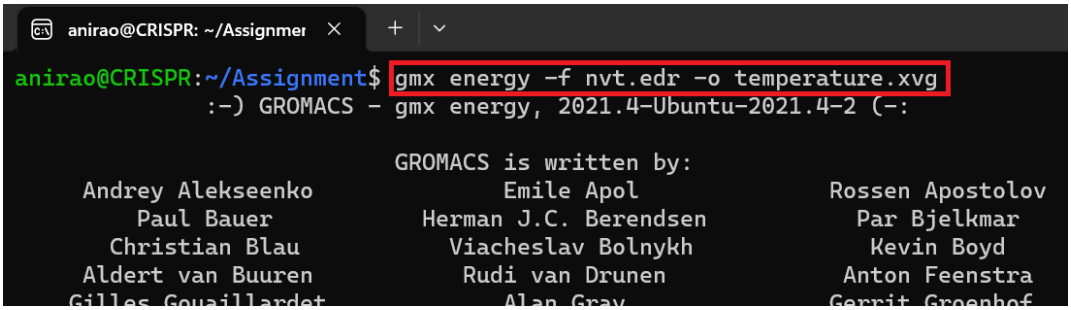


Figure 14: Command to analyse temperature during NVT equilibration

The average temperature is found to be **299.874 K**.

```
  1  Bond             2  Angle            3  Proper-Dih.      4  Ryckaert-Bell.
  5  LJ-14            6  Coulomb-14       7  LJ-(SR)          8  Disper.-corr.
  9  Coulomb-(SR)    10  Coul.-recip.    11  Position-Rest.  12  Potential
 13  Kinetic-En.     14  Total-Energy    15  Conserved-En.   16  Temperature
 17  Pres.-DC        18  Pressure        19  Constr.-rmsd    20  Vir-XX
 21  Vir-XY          22  Vir-XZ          23  Vir-YX          24  Vir-YY
 25  Vir-YZ          26  Vir-ZX          27  Vir-ZY          28  Vir-ZZ
 29  Pres-XX         30  Pres-XY         31  Pres-XZ         32  Pres-YX
 33  Pres-YY         34  Pres-YZ         35  Pres-ZX         36  Pres-ZY
 37  Pres-ZZ         38  #Surf*SurfTen   39  T-Protein       40  T-non-Protein
 41  Lamb-Protein                        42  Lamb-non-Protein

16
0
Last energy frame read 100 time  100.000

Statistics over 50001 steps [ 0.0000 through 100.0000 ps ], 1 data sets
All statistics are over 501 points

Energy                      Average   Err.Est.       RMSD  Tot-Drift
-------------------------------------------------------------------------------
Temperature                 299.874       0.25     3.0831    1.79987  (K)

GROMACS reminds you: "Make the Floor Burn" (2 Unlimited)
```

Figure 15: Average temperature during NVT equilibration

We can plot the temperature using Python.

```
In [1]:
import numpy as np
import matplotlib.pyplot as plt

x,y = np.loadtxt("temperature.xvg",comments=["@","#"],unpack=True)

plt.figure(dpi=100)
plt.plot(x,y)
plt.xlabel("Time (ps)")
plt.ylabel("Temperature (K)");
```

Figure 16: Python code to plot the temperature during NVT equilibration

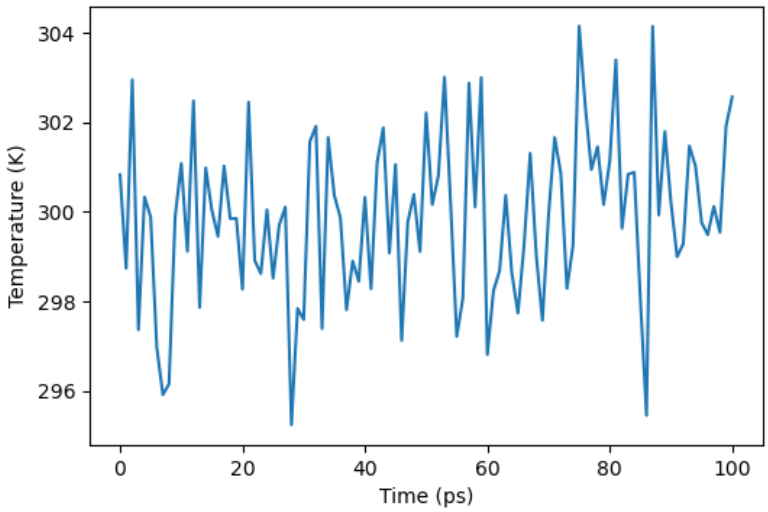


Figure 17: Temperature as a function of time during NVT equilibration

10. To run NPT equilibration for 150 ps, we have to modify the `npt.mdp` file. We will keep `dt` = 0.002 ps constant and change `nsteps` from 50000 to 75000 in order to run the simulation for 150 ps.

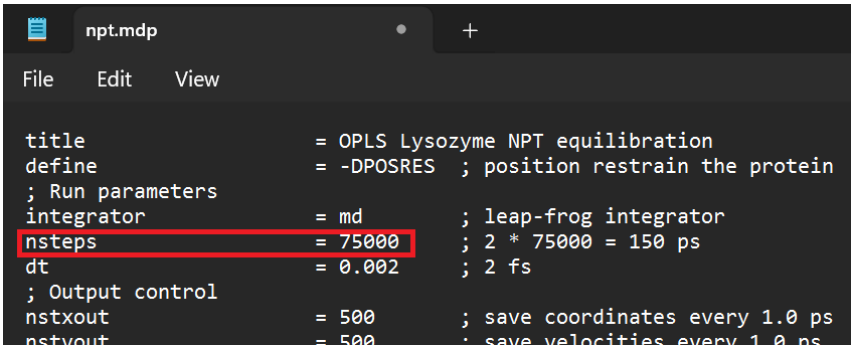


Figure 18: The edited `npt.mdp` file to run the NPT equilibration for 150 ps

The desired force constant is 1000 kJ/(mol nm^2). This is the already existing force constant in the position restraint `posre.itp` file. Hence, it does not need to be edited.

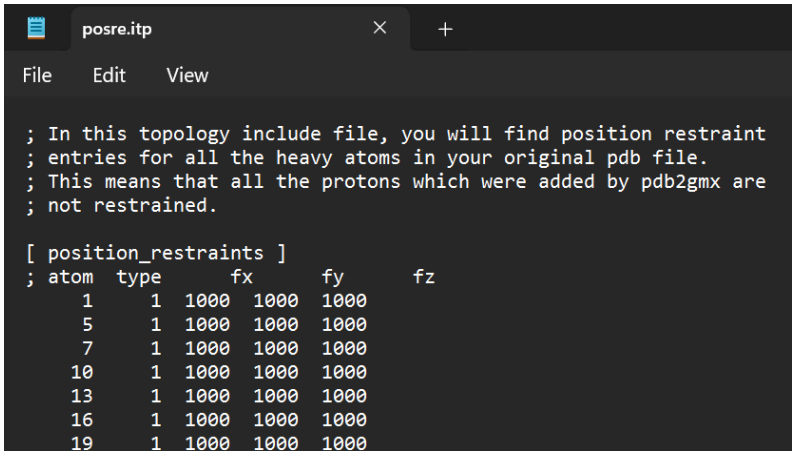


Figure 19: Force constants in the `posre.itp` file are already set to 1000 kJ/(mol nm^2)

To run the NPT equilibration, we have to first generate the `.tpr` file. We have to run the command

```
gmx grompp -f npt.mdp -c nvt.gro -r nvt.gro -t nvt.cpt -p
topol.top -o npt.tpr
```

`-t` specifies the trajectory as the checkpoint file produced by the NVT equilibration step.

After the `.tpr` file is generated, we can run the equilibration using the command

```
gmx mdrun -v -deffnm npt
```

This will perform NPT equilibration for 150 ps.

11. To run the MD simulation for 1 ns (= 1000 ps), we have to first create the `.tpr` file using `grompp`, specifying the `.mdp` file for the simulation.

```
gmx grompp -f md.mdp -c npt.gro -t npt.cpt -p topol.top -o
md_0_1.tpr
```

We do not specify any file with the `-r` flag as we are letting go of all position restraints.

Next, we can run the simulation with the command

```
gmx mdrun -v -deffnm md_0_1
```

Once the simulation is complete, we have to remove the PBC from the trajectory. To do this, we have to run the command

```
gmx trjconv -s md_0_1.tpr -f md_0_1.xtc -o md_0_1_noPBC.xtc -
pbc mol -center
```

`-s` specifies the structure file.

`-f` specifies the input trajectory file, which has the PBC.

`-o` specifies the output trajectory file, with removed PBC.

`-pbc` specifies the periodic boundary conditions used. In this case, `mol` refers to the fact that the centre of mass of all the molecules is within the box.

`-center` centers the atoms in the box.

On being prompted, we have to select 1 ("Protein") as the group to be centered and 0 ("System") for output.

To find the distance between the N and C termini of the protein during the course of the simulation, we have to first create an index file (`.ndx`) containing the atom positions we are interested in. From the `md_0_1.gro` file, we can see that the N terminus is at position 1 while the C terminus is at position 1958.

We can create an index file by running the command

```
gmx select -f md_0_1_noPBC.xtc -on termini.ndx -select "atomnr 1 1958"
```

`-f` specifies the trajectory file.

`-on` specifies the output index file.

`-select` is user to select the required atoms. In this case, we use the keyword `atomnr` to denote atom number.

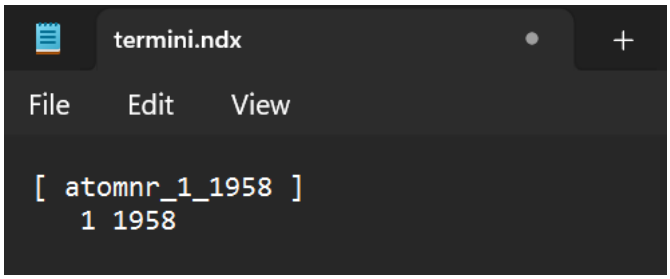


Figure 20: The index (`.ndx`) file generated by `select`

Next, we can find the distance between the two terminal atoms as a function of time using the command

```
gmx distance -f md_0_1_noPBC.xtc -s md_0_1.tpr -n termini.ndx -oall distance.xvg
```

`-f` specifies the trajectory file.

`-s` specifies the structure file.

`-n` specifies the index file.

`-oall` specifies that the pairwise distance has to be calculated for all pairs in the index file and then output to an `.xvg` file.

On being prompted, we type 0 (to denote the first and only pair in the index file) and hit Enter. We then press Ctrl + D to stop giving input. The distance is then calculated.

The distance between the two terminal atoms can be visualised in Python.

```
In [1]:
import numpy as np
import matplotlib.pyplot as plt

x,y = np.loadtxt("distance.xvg",comments=["@","#"],unpack=True)

plt.figure(dpi=100)
plt.plot(x,y)
plt.xlabel("Time (ps)")
plt.ylabel("Distance between N and C termini");
```

Figure 21: Python code to plot the distance between N and C termini during the simulation

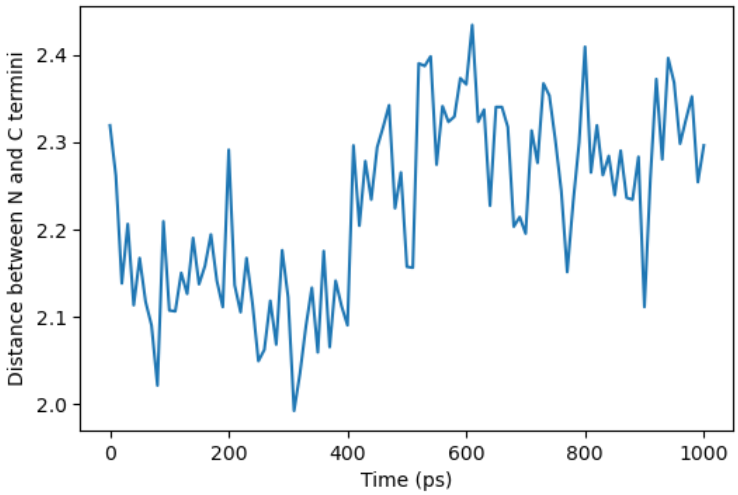


Figure 22: Distance between the N and C termini during the simulation, as a function of time

The average distance is 2.22681 nm, and the standard deviation of the distance is 0.10486 nm.

```
atomnr_1_1958:
  Number of samples:  101
  Average distance:   2.22681  nm
  Standard deviation: 0.10486  nm

GROMACS reminds you: "Science and eve
```

Figure 23: Average and standard deviation of the distance between the N and C termini during the simulation