BT5420: **Computer Simulations of Biomolecular Systems**

Project Report

Anirudh Rao

BE21B004

Contents

Objective 2
Reason for choosing the paper 2
Description of results being reproduced in this project 3
Details of simulation provided in the paper 4
Details of simulation performed in this project 5
Comparison of project results with original results 15
Acknowledgements 18
Appendix 18
Ramachandran Plot 26

The paper I have chosen for this project is

Tardigrade Secretory-Abundant Heat-Soluble Protein Has a Flexible β-Barrel Structure in Solution and Keeps This Structure in Dehydration
Kazuhisa Miyazawa et al. The Journal of Physical Chemistry B 2021 125 (32), 9145-9154. DOI: 10.1021/acs.jpcb.1c04850

Objective

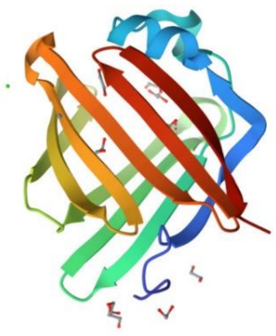
The objective of the paper was to simulate the secretory-abundant heat-soluble (SAHS) protein found in the tardigrade *Ramazzottius varieornatus*. Tardigrades are eight-legged, microscopic organisms that are capable of surviving extremely harsh environmental conditions. The protein being studied is hypothesised to be involved in helping the organism overcome desiccation-related stresses. By simulating the protein in solution and then under dehydrating conditions, the authors show that the protein maintains its characteristic structure, indicating its possible role in desiccation response.

A tardigrade, also known as a 'water bear'

Reason for choosing the paper

I have been fascinated by tardigrades and their extremotolerant behaviour ever since I heard about them in school. Due to their unique properties of surviving under desiccation stress, thermal stress and radiation stress, they are ideal candidates for organisms to use on space missions. As this paper demonstrates a molecular mechanism by which tardigrades can tolerate low water conditions, I decided to replicate their results.

In this paper, the authors did not use GROMACS for their simulations. Thus, I thought it would be a good challenge to translate the simulations they performed in another software to GROMACS. Replicating the dehydrating conditions required me to learn a bit of shell scripting, which was very interesting.



RvSAHS1: The secretory-abundant heat-soluble protein being simulated in this project (Source: RCSB PDB)

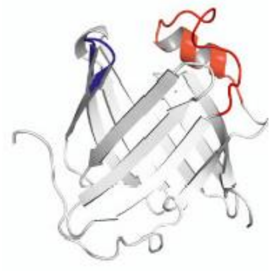
Description of results being reproduced in this project

Result #1

The authors show that under dehydrating conditions, the β-barrel of the protein maintains its structure. This can be shown by analysing the protein's root mean square fluctuation (RMSF). RMSF measures the average deviation of the C_α atom of each residue over time from its time-averaged position. If the β-barrel's residues have a low RMSF, it indicates that the structure is maintained under dehydration.

Result #2

The authors show that under dehydrating conditions, the N-terminal residue interacts with negatively-charged residues near the β-barrel. If the distance between these residues is small during dehydration, it indicates close interaction. This interaction could play a molecular role in stabilising the protein. The authors consider minimising the distance between the residues as the same as maximising the probability of contact between the residues.



Result #3

The authors show that the distance between the α helices (shown in red on the left) and the β sheets (shown in blue on the left) that form the "entrance" region of the protein starts reducing under dehydrating conditions. By coming closer, the strength of the hydrogen bonds in the entrance region are hypothesised to increase, providing greater stability during dehydration stress.

Overall, these results demonstrate the importance of this protein in helping tardigrades overcome desiccation-related stresses.

Details of simulation provided in the paper

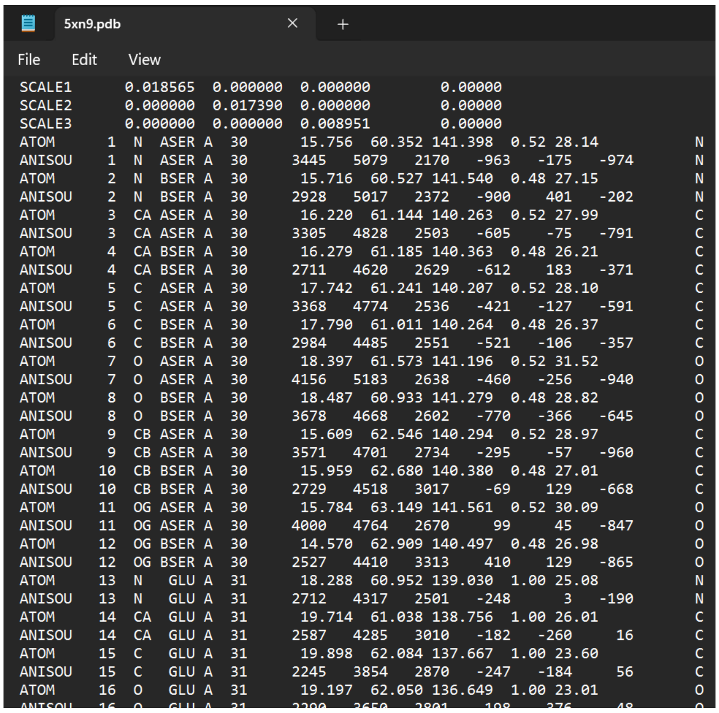
PDB ID of protein	5XN9
Part of protein used for simulation	Chain A, Ala20 to Ser169
Software used to edit protein	UCSF Chimera
Software used for simulation	GEMB
Box type	Cubic
Box dimensions	6.836 × 6.836 × 6.836 nm
Counter-ions	3 Na^+
Force field	AMBER14SB
Water model	TIP3P
Cutoff distance	1.2 nm
Calculation of electrostatic interactions	Particle Mesh Ewald (PME)
Integrator	r-RESPA
Integration time steps	4.0 fs for solvent interactions
	2.0 fs for nonbonding interactions
	0.5 fs for other interactions
Temperature	300 K
Pressure	0.1 MPa = 1 bar
Thermostat	Nosé-Hoover
Barostat	Andersen
Equilibration time	20 ns
Solution simulation time	300 ns
Dehydration simulation time	200 ns
Dehydration simulation strategy	Randomly remove 50 water molecules every 1 ns with barostat turned off

The above details are explicitly mentioned in the paper. Other simulation details will be assumed to be the same as the standard conditions used in GROMACS.

Details of simulation performed in this project

Step 1 – Download the `.pdb` file and inspect it

The `.pdb` file for the protein being simulated was downloaded from the RCSB Protein Data Bank. The PDB ID of the protein is 5XN9. The structure of the protein was determined by X-ray diffraction. The structure contains the data of two chains – A and B. The coordinates of the first 30 residues and the last 2 residues are missing. According to the depositors of this protein structure, the first residue (Ser) is an artifact from the purification process and has to be ignored. Every set of atomic coordinates also has a set of anisotropic B-factors (`ANISOU`) which can be ignored for our purposes. The file also contains the coordinates of non-protein molecules – Zn^{2+}, ethanediol, acetic acid, Cl^-, Mg^{2+}, and water.



```
SCALE1      0.018565  0.000000  0.000000        0.00000
SCALE2      0.000000  0.017390  0.000000        0.00000
SCALE3      0.000000  0.000000  0.008951        0.00000
ATOM      1  N  ASER A  30      15.756  60.352 141.398  0.52 28.14           N
ANISOU    1  N  ASER A  30     3445   5079   2170   -963   -175   -974       N
ATOM      2  N  BSER A  30      15.716  60.527 141.540  0.48 27.15           N
ANISOU    2  N  BSER A  30     2928   5017   2372   -900    401   -202       N
ATOM      3  CA ASER A  30      16.220  61.144 140.263  0.52 27.99           C
ANISOU    3  CA ASER A  30     3305   4828   2503   -605    -75   -791       C
ATOM      4  CA BSER A  30      16.279  61.185 140.363  0.48 26.21           C
ANISOU    4  CA BSER A  30     2711   4620   2629   -612    183   -371       C
ATOM      5  C  ASER A  30      17.742  61.241 140.207  0.52 28.10           C
ANISOU    5  C  ASER A  30     3368   4774   2536   -421   -127   -591       C
ATOM      6  C  BSER A  30      17.790  61.011 140.264  0.48 26.37           C
ANISOU    6  C  BSER A  30     2984   4485   2551   -521   -106   -357       C
ATOM      7  O  ASER A  30      18.397  61.573 141.196  0.52 31.52           O
ANISOU    7  O  ASER A  30     4156   5183   2638   -460   -256   -940       O
ATOM      8  O  BSER A  30      18.487  60.933 141.279  0.48 28.82           O
ANISOU    8  O  BSER A  30     3678   4668   2602   -770   -366   -645       O
ATOM      9  CB ASER A  30      15.609  62.546 140.294  0.52 28.97           C
ANISOU    9  CB ASER A  30     3571   4701   2734   -295    -57   -960       C
ATOM     10  CB BSER A  30      15.959  62.680 140.380  0.48 27.01           C
ANISOU   10  CB BSER A  30     2729   4518   3017    -69    129   -668       C
ATOM     11  OG ASER A  30      15.784  63.149 141.561  0.52 30.09           O
ANISOU   11  OG ASER A  30     4000   4764   2670     99     45   -847       O
ATOM     12  OG BSER A  30      14.570  62.909 140.497  0.48 26.98           O
ANISOU   12  OG BSER A  30     2527   4410   3313    410    129   -865       O
ATOM     13  N   GLU A  31      18.288  60.952 139.030  1.00 25.08           N
ANISOU   13  N   GLU A  31     2712   4317   2501   -248      3   -190       N
ATOM     14  CA  GLU A  31      19.714  61.038 138.756  1.00 26.01           C
ANISOU   14  CA  GLU A  31     2587   4285   3010   -182   -260     16       C
ATOM     15  C   GLU A  31      19.898  62.084 137.667  1.00 23.60           C
ANISOU   15  C   GLU A  31     2245   3854   2870   -247   -184     56       C
ATOM     16  O   GLU A  31      19.197  62.050 136.649  1.00 23.01           O
```

The original `.pdb` file

Step 2 – Clean the `.pdb` file

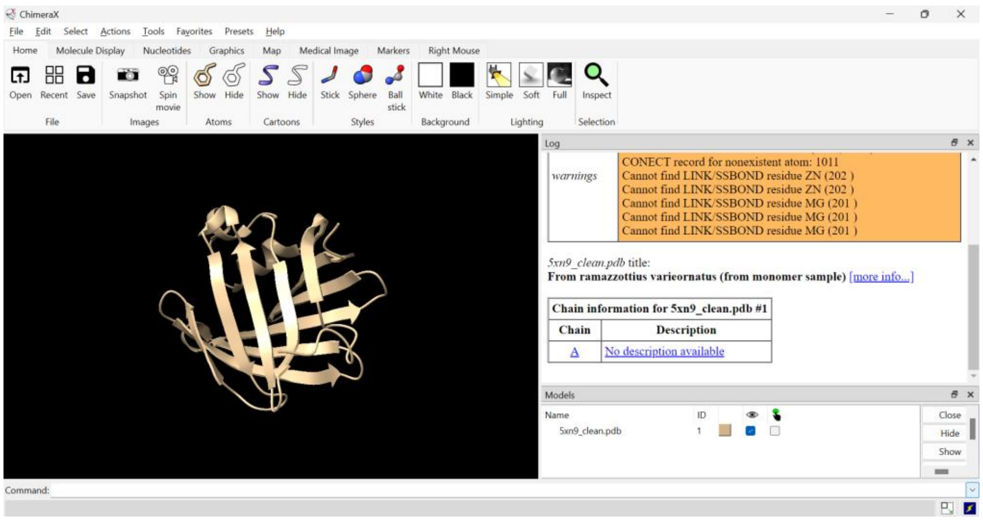
We have to remove the coordinates of residues in Chain B. We also have to remove the anisotropic B-factors and the heteroatom coordinates. To do this, we open Ubuntu and run the command

```
grep -v 'B\|ANISOU\|HETATM' 5xn9.pdb > 5xn9_clean.pdb
```

The symbol `\|` denotes 'OR'. The flag `-v` displays those lines that do not contain matches to the specified words.

Step 3 – Fill the missing residues

To fill the missing residues in the N and C termini, we use the same software that the authors used – UCSF Chimera, which can be downloaded from here. After opening the software, we navigate to Home > Open and open the `5xn9_clean.pdb` file.



```
warnings  CONECT record for nonexistent atom: 1011
          Cannot find LINK/SSBOND residue ZN (202 )
          Cannot find LINK/SSBOND residue ZN (202 )
          Cannot find LINK/SSBOND residue MG (201 )
          Cannot find LINK/SSBOND residue MG (201 )
          Cannot find LINK/SSBOND residue MG (201 )

5xn9_clean.pdb title:
From ramazzottius varieornatus (from monomer sample) [more info...]
```

Chain information for 5xn9_clean.pdb #1	
Chain	**Description**
A	No description available

Name	ID			
5xn9_clean.pdb	1		☑	☐

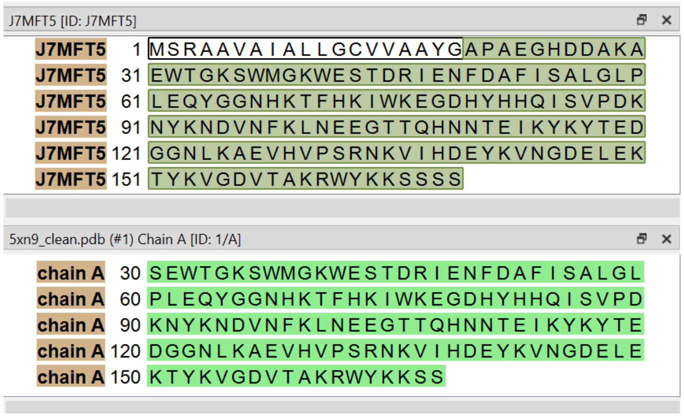
ChimeraX
File Edit Select Actions Tools Favorites Presets Help
Home Molecule Display Nucleotides Graphics Map Medical Image Markers Right Mouse
Open Recent Save Snapshot Spin movie Show Hide Show Hide Stick Sphere Ball stick White Black Simple Soft Full Inspect
File Images Atoms Cartoons Styles Background Lighting Selection
Log
Models
Close Hide Show
Command:

5XN9 in UCSF Chimera

We can build the missing coordinates by providing the exact protein sequence. This can be done by running the command (in the command line at the bottom)

```
open uniprot:J7MFT5
```

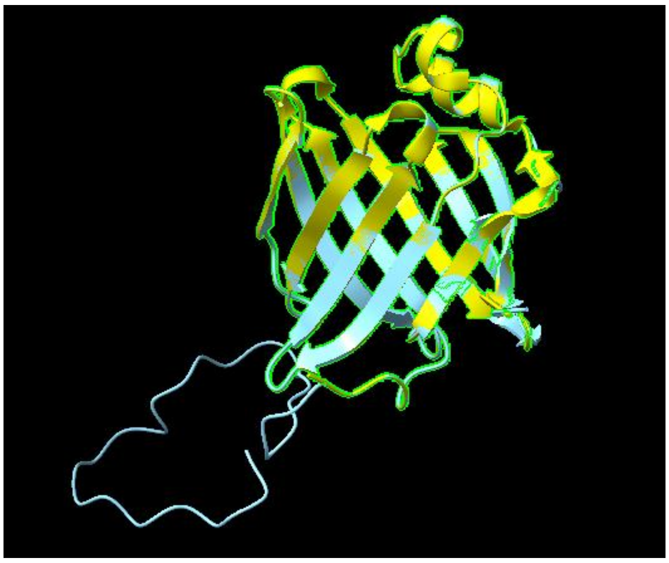
J7MFT5 is the UniProt ID of the same protein. In the popup window that appears, click 'Chain' and close the window. Then, navigate to Tools > Sequence > Show Sequence Viewer and click on 'Show' in the window that appears. The missing residues can now be seen.

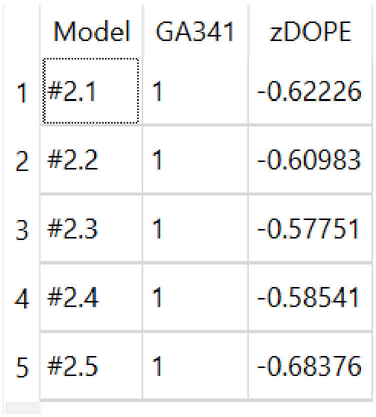


```
J7MFT5 [ID: J7MFT5]
J7MFT5   1 MSRAAVAIALLGCVVAAYGAPAEGHDDAKA
J7MFT5  31 EWTGKSWMGKWESTDRIENFDAFISALGLP
J7MFT5  61 LEQYGGNHKTFHKIWKEGDHYHHQISVPDK
J7MFT5  91 NYKNDVNFKLNEEGTTQHNNTEIKYKYTED
J7MFT5 121 GGNLKAEVHVPSRNKVIHDEYKVNGDELEK
J7MFT5 151 TYKVGDVTAKRWYKKSSSS

5xn9_clean.pdb (#1) Chain A [ID: 1/A]
chain A  30 SEWTGKSWMGKWESTDRIENFDAFISALGL
chain A  60 PLEQYGGNHKTFHKIWKEGDHYHHQISVPD
chain A  90 KNYKNDVNFKLNEEGTTQHNNTEIKYKYTE
chain A 120 DGGNLKAEVHVPSRNKVIHDEYKVNGDELE
chain A 150 KTYKVGDVTAKRWYKKSS
```

The complete and incomplete protein sequences

To actually fill the residues, we navigate to Tools > Sequence > Model Loops and click ‘OK’ in the window that appears. This uses the existing structure of `5xn9_clean.pdb` as a template for the sequence provided by UniProt. By default, this generates 5 possible structures. The structure with a zDOPE score closest to –1 is taken as the final structure. In this case, it was structure #2.5. The authors, however, generated 20 structures and ran MD simulations on all of them.



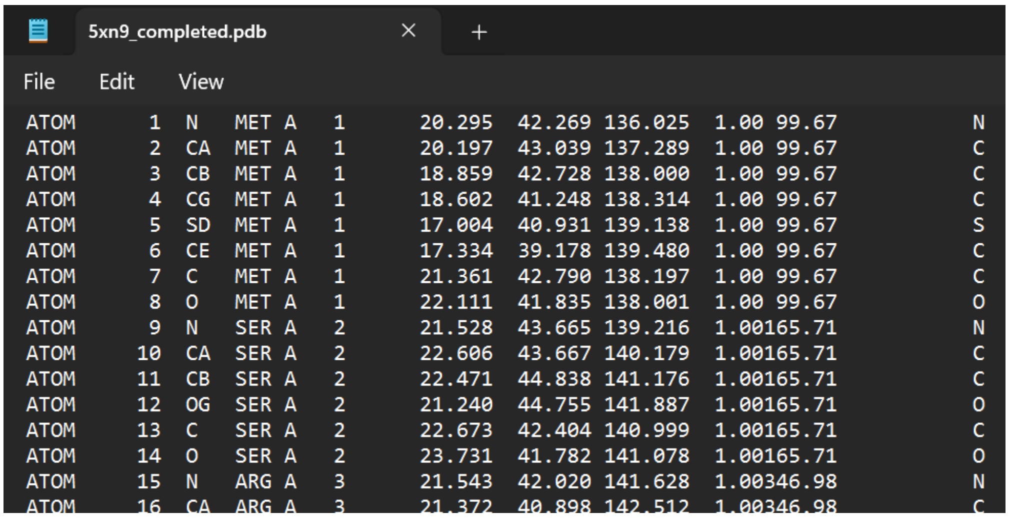


	Model	GA341	zDOPE
1	#2.1	1	-0.62226
2	#2.2	1	-0.60983
3	#2.3	1	-0.57751
4	#2.4	1	-0.58541
5	#2.5	1	-0.68376

The original (yellow) and completed (blue) structures, and the results of residue filling

We save the completed coordinates by running the command

```
save 5xn9_completed.pdb models #2.5
```



```
ATOM      1  N   MET A   1      20.295  42.269 136.025  1.00 99.67           N
ATOM      2  CA  MET A   1      20.197  43.039 137.289  1.00 99.67           C
ATOM      3  CB  MET A   1      18.859  42.728 138.000  1.00 99.67           C
ATOM      4  CG  MET A   1      18.602  41.248 138.314  1.00 99.67           C
ATOM      5  SD  MET A   1      17.004  40.931 139.138  1.00 99.67           S
ATOM      6  CE  MET A   1      17.334  39.178 139.480  1.00 99.67           C
ATOM      7  C   MET A   1      21.361  42.790 138.197  1.00 99.67           C
ATOM      8  O   MET A   1      22.111  41.835 138.001  1.00 99.67           O
ATOM      9  N   SER A   2      21.528  43.665 139.216  1.00165.71           N
ATOM     10  CA  SER A   2      22.606  43.667 140.179  1.00165.71           C
ATOM     11  CB  SER A   2      22.471  44.838 141.176  1.00165.71           C
ATOM     12  OG  SER A   2      21.240  44.755 141.887  1.00165.71           O
ATOM     13  C   SER A   2      22.673  42.404 140.999  1.00165.71           C
ATOM     14  O   SER A   2      23.731  41.782 141.078  1.00165.71           O
ATOM     15  N   ARG A   3      21.543  42.020 141.628  1.00346.98           N
ATOM     16  CA  ARG A   3      21.372  40.898 142.512  1.00346.98           C
```

The completed `.pdb` file

Step 4 – Remove the signal peptide

The authors mention that Met1–Gly19 acts as a signal peptide and gets cleaved off after the protein is secreted. Thus, Ala20–Ser169 is employed as the functional protein by simply deleting the coordinates of Met1–Gly19 from `5xn9_completed.pdb` and saving this new file as `5xn9_edited.pdb`.

Step 5 – Create a force field-compliant topology

We run the command

```
gmx pdb2gmx -f 5xn9_edited.pdb -o 5xn9.gro -water tip3p -ignh
```

When prompted, we choose the AMBER99SB force field, which is the closest to the AMBER14SB force field used by the authors. We can do this based on the fact that 14SB is derived from 99SB. The files `5xn9.gro`, `topol.top` and `posre.itp` are generated.

Step 6 – Solvation

First, we define the simulation box by running the command

```
gmx editconf -f 5xn9.gro -o 5xn9_box.gro -c -bt cubic -box 6.836
```

This creates a cubic box of the same dimensions as the one mentioned in the paper. Then, we solvate the system by running the command

```
gmx solvate -cp 5xn9_box.gro -cs spc216.gro
-o 5xn9_solvated.gro -p topol.top
```

This adds 9687 water molecules to the box. In the paper, the authors added an average of 9522 water molecules across their 20 simulations of the same protein.

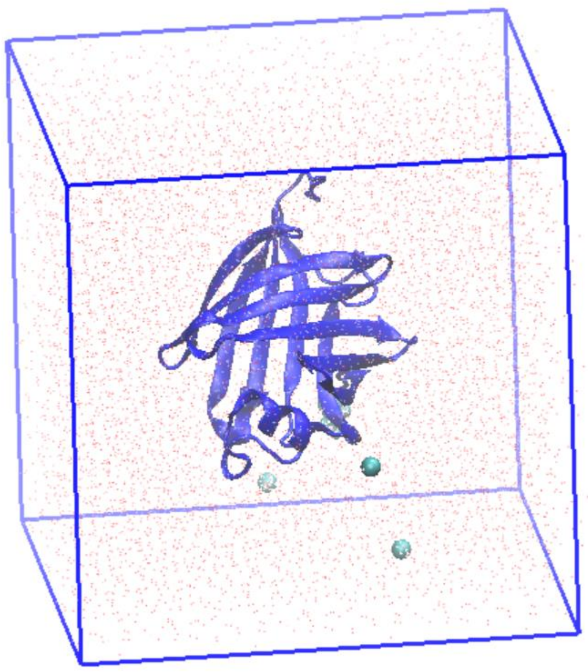
Step 7 – Neutralisation

We modify the `ions.mdp` file to change the cutoff distance from 1.0 nm to 1.2 nm. We do not change `coulombtype` from `cutoff` to `PME` as `PME` requires that the system should not have any net charge. We save this file as `new_ions.mdp`, which can be found in the Appendix. We then run the commands

```
gmx grompp -f new_ions.mdp -c 5xn9_solvated.gro -p topol.top
-o ions.tpr
```

```
gmx genion -s ions.tpr -o 5xn9_neutral.gro -p topol.top
-pname NA -nname CL -neutral
```

When prompted, we select Group 13 – SOL as the group to replace ions with. This command replaces 3 water molecules with 3 Na^+ ions, just like in the paper.


Solvated and neutralised protein in a cubic box

Step 8 – Energy minimisation

This step was not explicitly carried out in the paper. We will do this to ensure that the system has no steric clashes or inappropriate geometry.

We modify the `minim.mdp` file to change the cutoff distance from 1.0 nm to 1.2 nm. `coulombtype` was already changed from `cutoff` to `PME` in the original `minim.mdp` file. We save this file as `new_minim.mdp`, which can be found in the Appendix.

We then run the commands

```
gmx grompp -f new_minim.mdp -c 5xn9_neutral.gro -p topol.top
-o em.tpr

gmx mdrun -v -deffnm em
```

Step 9 – NVT equilibration

We carry out NVT equilibration for 100 ps. To replicate the multiple time steps used by the authors, we change `dt` to 0.5 fs. Accordingly, we change `nsteps` to 200000. In addition to these changes to the original `nvt.mdp` file, we also specify the multiple time steps we wish to use.

Unfortunately, GROMACS only allows us to specify one additional time step, whereas the paper uses two additional time steps. Thus, for all non-bonded interactions, we specify a time step of 2.0 fs, instead of 4.0 fs for solvent non-bonded interactions and 2.0 fs for other non-bonded interactions.

To ensure that the neighbour list (`nstlist`) gets updated at the same frequency as the original `nvt.mdp` file, we change the number of steps from 10 to 40 to retain the frequency as 20 fs.

We change the cutoff distance from 1.0 nm to 1.2 nm. We change the thermostat from `v-rescale` to `nose-hoover`.

The modified file was saved as `new_nvt.mdp`, which can be found in the Appendix.

We then run the commands

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

This runs the NVT equilibration for 100 ps.

Step 10 – NPT equilibration

The `new_npt.mdp` file is derived from the `new_nvt.mdp` file. In the `new_npt.mdp` file, pressure coupling is turned on. As the Andersen barostat used by the authors is not present in GROMACS, we use the closely related Parrinello-Rahman barostat, which is already present in the original `npt.mdp` file. `new_npt.mdp` can be found in the Appendix.

We then run the commands

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

This runs the NPT equilibration for 100 ps.

Step 11 – Production run

We will simulate the protein for 1.5 ns in solution and 1 ns under dehydrating conditions. This is essentially a "scaled-down" version of the simulation run by the authors (who ran 300 ns for solution and 200 ns for dehydration). The MD run in solution requires us to modify the `nsteps`, multiple time stepping, cutoff distances and thermostat in the original `md.mdp` file. We also have to modify the time constant for the thermostat from 0.1 ps to 0.4 ps to prevent a fatal error. This modified file is renamed as `new_md_soln.mdp` and can be found in the Appendix.

For the dehydration simulation, the authors randomly removed 50 water molecules every 1 ns over a course of 200 ns. If we scaled this to a 1 ns simulation, we would have to remove 50 water molecules every 5 ps, which is too frequent.

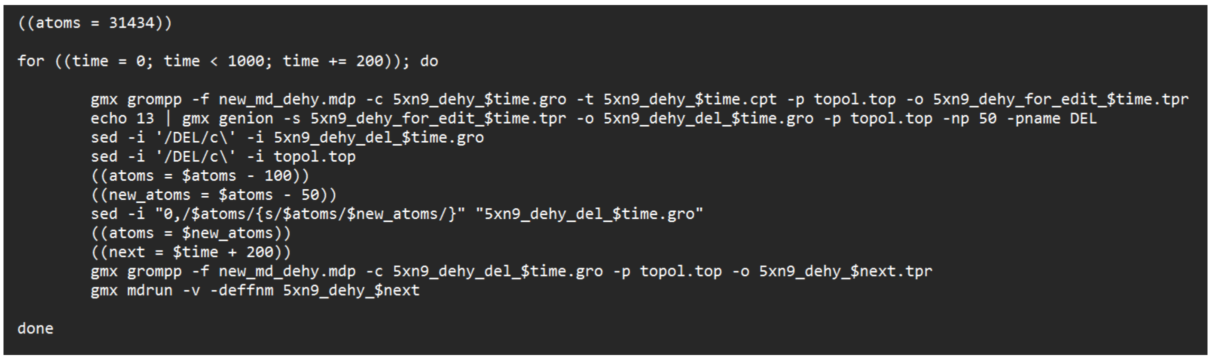
The course instructor, Dr Hamsa Priya Mohana Sundaram, suggested that I should do this random removal 5 times over the 1 ns simulation, i.e., randomly remove 50 water molecules every 200 ps. For the dehydration simulation, the `new_md_soln.mdp` file was modified such that the barostat was turned off (as the density will not remain constant). This modified file was renamed as `new_md_dehy.mdp`, and can be found in the Appendix.

To perform the production run, including the solution and dehydration simulations, a simple shell script, called `production.sh`, was written. This can be found in the Appendix.

First, the script executed a 1.5 ns simulation of the protein in solution by running the commands

```
gmx grompp -f new_md_soln.mdp -c npt.gro -t npt.cpt
        -p topol.top -o 5xn9_dehy_0.tpr

gmx mdrun -v -deffnm 5xn9_dehy_0
```

Then, the script executed five 200 ps simulations, each starting with the removal of 50 random water molecules. The removal was carried out using `genion` and specifying that ‘`DEL`’ should replace the solvent molecules (Group 13).



```
((atoms = 31434))

for ((time = 0; time < 1000; time += 200)); do

        gmx grompp -f new_md_dehy.mdp -c 5xn9_dehy_$time.gro -t 5xn9_dehy_$time.cpt -p topol.top -o 5xn9_dehy_for_edit_$time.tpr
        echo 13 | gmx genion -s 5xn9_dehy_for_edit_$time.tpr -o 5xn9_dehy_del_$time.gro -p topol.top -np 50 -pname DEL
        sed -i '/DEL/c\' -i 5xn9_dehy_del_$time.gro
        sed -i '/DEL/c\' -i topol.top
        ((atoms = $atoms - 100))
        ((new_atoms = $atoms - 50))
        sed -i "0,/$atoms/{s/$atoms/$new_atoms/}" "5xn9_dehy_del_$time.gro"
        ((atoms = $new_atoms))
        ((next = $time + 200))
        gmx grompp -f new_md_dehy.mdp -c 5xn9_dehy_del_$time.gro -p topol.top -o 5xn9_dehy_$next.tpr
        gmx mdrun -v -deffnm 5xn9_dehy_$next

done
```

Shell script for dehydration

The shell script was run using the command

```
./production.sh
```

This performed the dehydration simulation over 1 ns.

Step 12 – Trajectory conversion and concatenation

Each of the `.xtc` files produced had PBC removed by running the command

```
gmx trjconv -s 5xn9_dehy_$time.tpr -f 5xn9_dehy_$time.xtc
     -o 5xn9_dehy_$time_no_pbc.xtc -pbc mol -center
```

`$time` takes the values `0, 200, 400, 600, 800, 1000`.

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

These converted trajectories will be used for the purpose of movie making.

For analysis, we only require the protein's trajectory. For this we first created an index file containing only the protein's atom numbers by running the command

```
gmx select -f 5xn9_dehy_0_no_pbc.xtc -on required_atoms.ndx
                -select "atomnr 1 to 2379"
```

Then, we re-ran `trjconv` with the command

```
gmx trjconv -f 5xn9_dehy_$time_no_pbc.xtc -o
5xn9_dehy_$time_protein.xtc -n required_atoms.ndx
```

`$time` takes the values `0, 200, 400, 600, 800, 1000`.

The trajectories corresponding only to the dehydration run were concatenated using the command

```
gmx trjcat -f 5xn9_dehy_200_protein.xtc
5xn9_dehy_400_protein.xtc 5xn9_dehy_600_protein.xtc
5xn9_dehy_800_protein.xtc 5xn9_dehy_1000_protein.xtc -o
5xn9_dehy_protein.xtc -cat
```

All trajectories, including the solution run, were concatenated using the command

```
gmx trjcat -f 5xn9_dehy_0_protein.xtc
5xn9_dehy_200_protein.xtc 5xn9_dehy_400_protein.xtc
5xn9_dehy_600_protein.xtc 5xn9_dehy_800_protein.xtc
5xn9_dehy_1000_protein.xtc -o 5xn9_full_protein.xtc -cat
```

Step 13 – Movie making

Each of the starting `.gro` files and their corresponding `.xtc` trajectory files with PBC removed were loaded separately into VMD. The protein was represented with Secondary Structure as the colouring method and with NewCartoon. Using VMD's inbuilt Movie Maker option, `.bmp` image files for each of the frames in the trajectory were created.

As VMD did not have a VideoMach plugin, Python's OpenCV library was used to convert the `.bmp` images into a `.mp4` video. The code used for this can be found in the Appendix.

Three movies were created – one of the protein in solution, one under dehydration and one combined movie. The movies created can be found in the Google Drive link provided at the end of this section.

Step 14 – Analysis

Three analyses that were performed in the paper were chosen and replicated.

Analysis #1 – Root mean square fluctuation

The root mean square fluctuation of the backbone C_α atoms of the protein was analysed during the dehydration run. The structure at the end of the solution run was used for comparison. The command used was

```
gmx rmsf -s 5xn9_dehy_0.gro -f 5xn9_dehy_protein.xtc
-o rmsf_dehydration.xvg
```

This data was plotted using Python and compared with the result given in the paper. The RMSF values for the residues that form the β-barrel of the protein were highlighted in the plot.

The comparison of results is discussed in the subsequent section. The code used for plotting is provided in the Appendix.

Analysis #2 – Distance between Ala20 and other residues

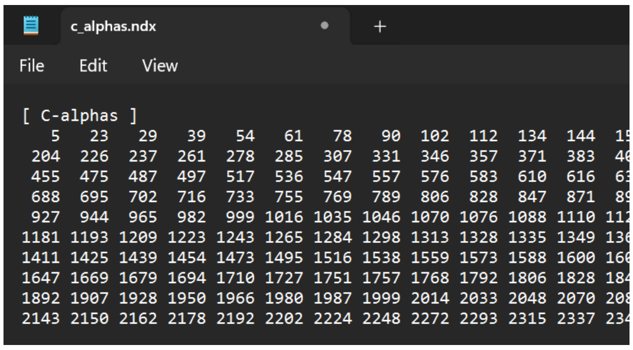
The distance between the C_α atom of Ala20 (at the N-terminus) and the C_α atoms of other residues in the protein was calculated as a function of time (during the dehydration run).

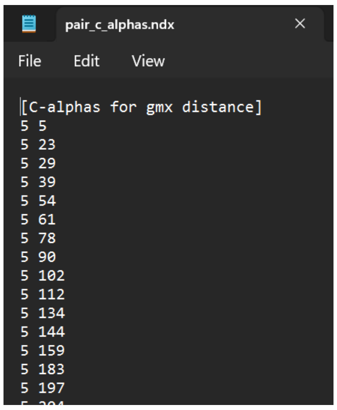
First, an index file containing the atom numbers of the C_α atoms was created using `make_ndx`.

```
gmx make_ndx -f 5xn9_dehy_0.gro -o c_alphas.ndx
```

`c_alphas.ndx` was manually edited to include only the C_α atom numbers.

Using Python, this was converted to a pairwise index file so that the distance between each of these atoms and Ala20 can be calculated. The code used is provided in the Appendix. The file containing these pairwise atom numbers was named `pair_c_alphas.ndx`.


```
[ C-alphas ]
    5   23   29   39   54   61   78   90  102  112  134  144  15
  204  226  237  261  278  285  307  331  346  357  371  383  40
  455  475  487  497  517  536  547  557  576  583  610  616  63
  688  695  702  716  733  755  769  789  806  828  847  871  89
  927  944  965  982  999 1016 1035 1046 1070 1076 1088 1110 112
 1181 1193 1209 1223 1243 1265 1284 1298 1313 1328 1335 1349 136
 1411 1425 1439 1454 1473 1495 1516 1538 1559 1573 1588 1600 160
 1647 1669 1679 1694 1710 1727 1751 1757 1768 1792 1806 1828 184
 1892 1907 1928 1950 1966 1980 1987 1999 2014 2033 2048 2070 208
 2143 2150 2162 2178 2192 2202 2224 2248 2272 2293 2315 2337 234
```


```
[C-alphas for gmx distance]
5 5
5 23
5 29
5 39
5 54
5 61
5 78
5 90
5 102
5 112
5 134
5 144
5 159
5 183
5 197
```

Index files for Ala20 distance calculation

To calculate the distances, we run the command

```
gmx distance -f 5xn9_dehy_protein.xtc -s 5xn9_dehy_0.gro
-n pair_c_alphas.ndx -oall distances.xvg
```

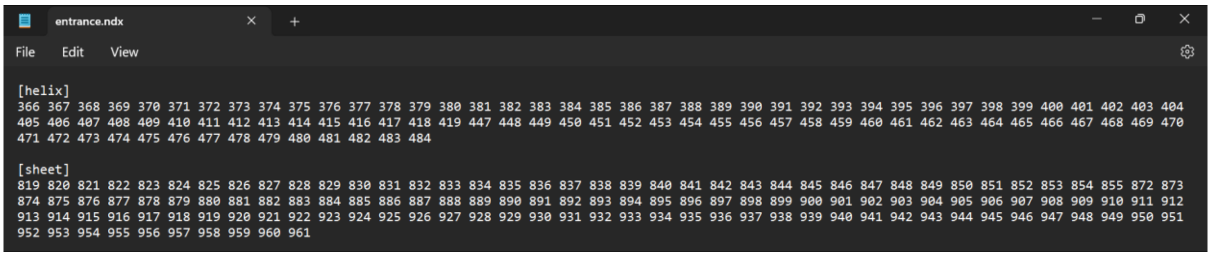
Using Python, the distances as a function of time were analysed for each residue and a probability that the residue is present at a distance lesser than a pre-defined threshold was computed. These "contact" probabilities were plotted as a function of residue number.

The comparison of results obtained here and in the original paper is discussed in the subsequent section. The code used for analysing and plotting is provided in the Appendix.

Analysis #3 – Distance between the α helices and the most fluctuated β sheets

The two α helices and the 5th and 6th β sheets form the "entrance" region of the protein. From the results of the RMSF analysis, we can see that these β sheets show high fluctuation during dehydration. Thus, we analyse the distance between the α helices and the β sheets that form the entrance, as a function of time.

First, we create an index file containing the atom numbers of all the backbone atoms of the groups of interest. This is done manually and saved as `entrance.ndx`.


```
[helix]
366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404
405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470
471 472 473 474 475 476 477 478 479 480 481 482 483 484

[sheet]
819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 872 873
874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912
913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951
952 953 954 955 956 957 958 959 960 961
```

Index file of entrance region backbone atoms

To compute the distance between the two groups, i.e., the distance between their centres of gravity, we run the command

```
gmx distance -f 5xn9_full_protein.xtc -s 5xn9_dehy_0.tpr
-n entrance.ndx -oall entrance_dist.xvg -select "cog of group
helix plus cog of group sheet"
```

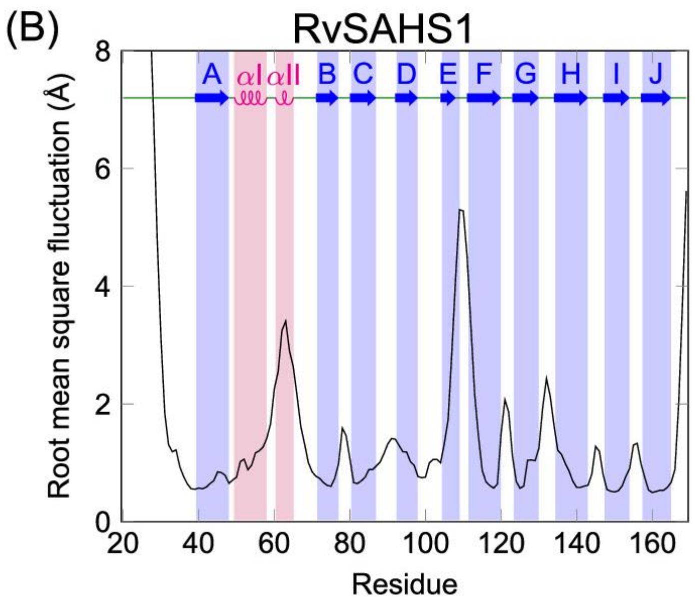
The calculated distances are plotted and analysed using Python. Trendlines for the distances when the protein is in solution and under dehydration are also plotted. The code used can be found in the Appendix. The results of the analysis are discussed in the subsequent section.

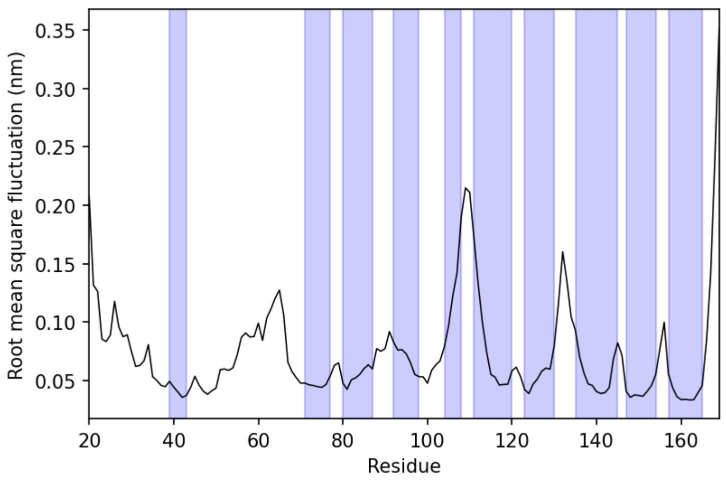
All files generated for and used during this project can be found at

https://drive.google.com/drive/folders/1yj0FsWfCNB4sMa5YvTV7w5M5v__yUcJK?usp=sharing

Comparison of project results with original results

Comparison #1 – Root mean square fluctuation

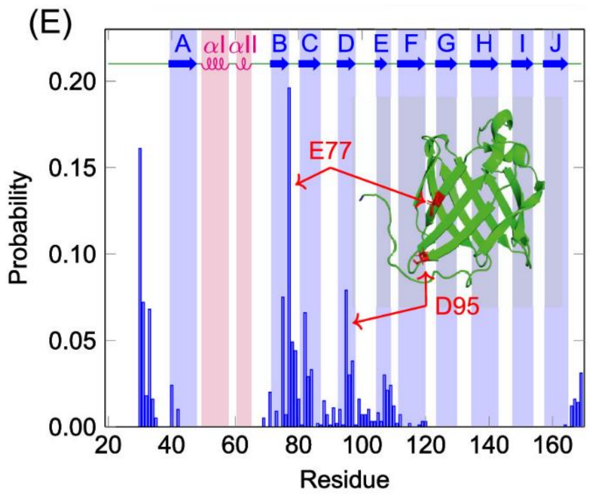


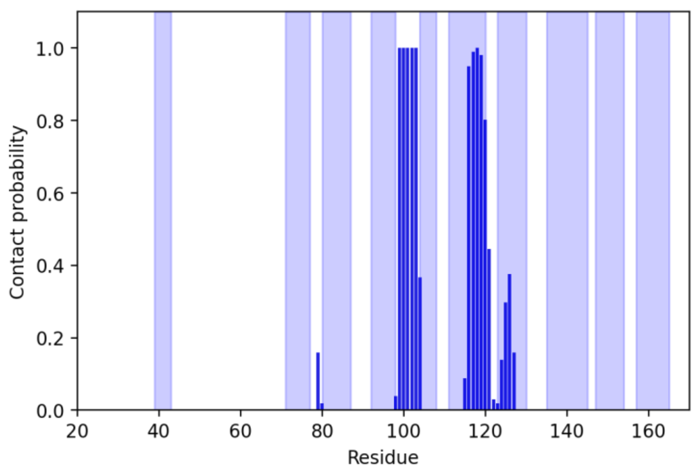


The figure on the left is from Fig. 1(B) of the paper, and the figure on the right is the one plotted using the results of this project. The vertical blue bands indicate the β sheets that form the barrel.

Quantitatively, the two plots are different as the simulation run in this project is only 1 ns whereas the authors performed a 200 ns simulation. However, the two plots are qualitatively very similar. We can clearly see that both plots have similar peaks for the same residues. It is also clear that the RMSF for the β-barrel takes minimum values, indicating its stability.

Comparison #2 – Distance between Ala20 and other residues





The figure on the left is from Fig. 2(E) of the paper, and the figure on the right is the one plotted using the results of this project. The vertical blue bands indicate the β sheets that form the barrel.

The figure created for this project does not reflect the results obtained in the paper. There are two primary reasons for this –

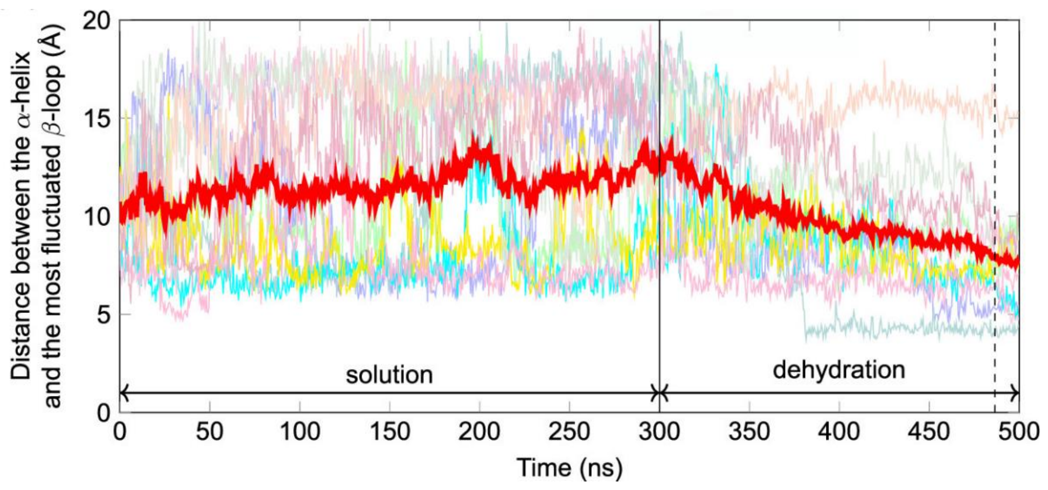
(i) The simulation performed in this project is only 1 ns long while the original paper carried out a 200 ns simulation.

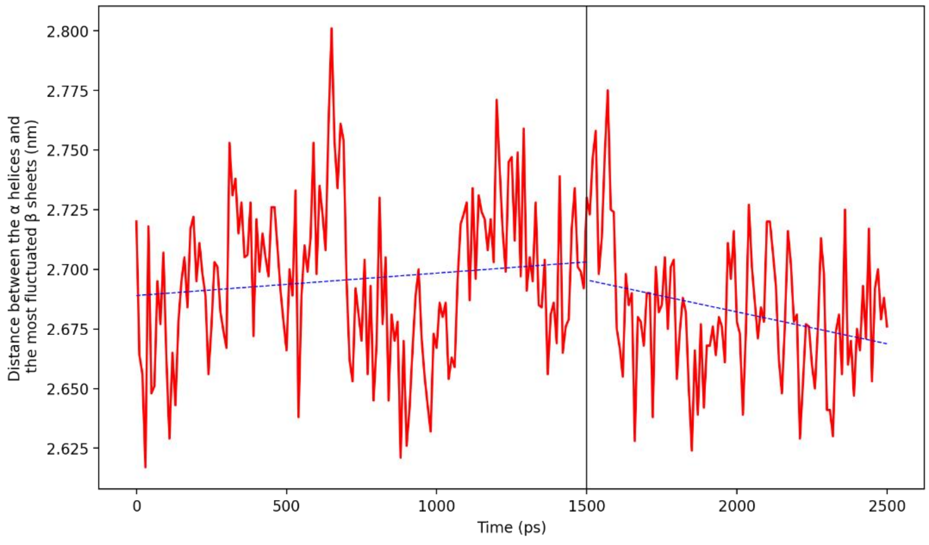
(ii) The threshold distance set to determine when a contact occurs is different. The authors used a 5 angstrom threshold but the distances obtained using this project did not capture such a fine degree of resolution. This project defines the threshold as the mean minus standard deviation of the minimum distances between each residue and Ala20.

However, we can analyse this project's result qualitatively. Our analysis shows that Glu119 and Asp120 have a high contact probability with Ala20 – 0.98 and 0.80 respectively. Both these negatively-charged residues are present in a β sheet. This is similar to the result obtained by the authors, who found that Glu77 and Asp95 have high contact probability.

This interaction between Ala20 and the β-barrel are indicative of a stabilisation mechanism used by the protein when it is subjected to dehydration.

Comparison #3 – Distance between the α helices and the most fluctuated β sheets





The figure on the top is from Fig. 3(C) of the paper, and the figure on the bottom is the one plotted using the results of this project. The black vertical line separates the solution and dehydration runs. The fluctuating red plot represents the calculated distance. The dashed blue line represents a trendline of the distance.

Since the simulation performed in this project is only 1.5 + 1 = 2.5 ns long, it does not reflect the exact results obtained in the paper. However, the trendline qualitatively resembles the authors' result. The overall plot resembles the "transition" region in the authors' plot. In solution, the distance between the α helices and the 5th and 6th β sheets remains relatively constant. When the protein is subjected to dehydration, this distance starts reducing, indicating that the entrance region of the protein is closing. The authors saw that this reduction in distance increases hydrogen bonding, improving the stability of the protein.

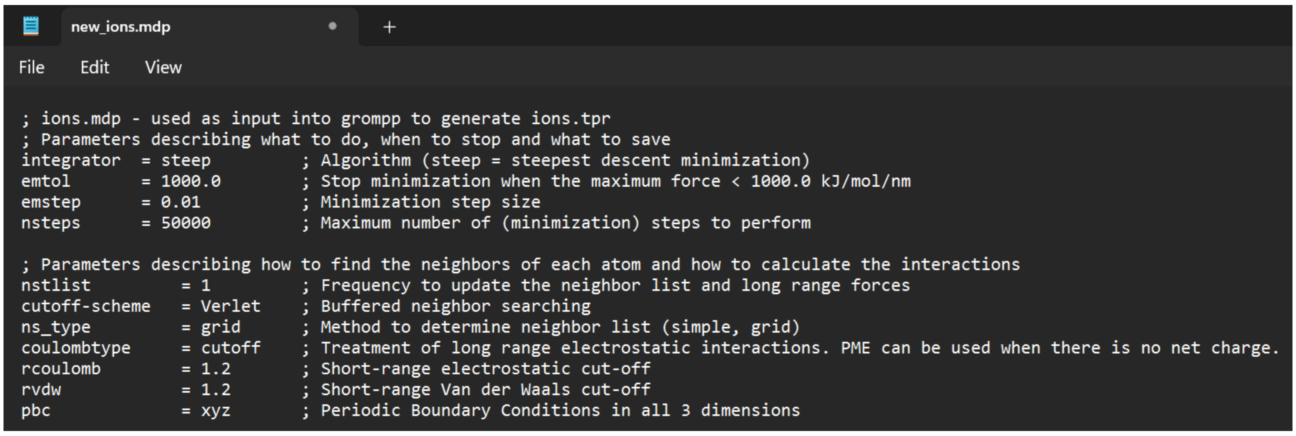
Thus, the results obtained in this project are qualitatively very similar to the ones obtained in the paper.

Acknowledgements

I would like to thank the course instructor Dr Hamsa Priya Mohana Sundaram for her valuable inputs during this project and throughout the course. I would also like to thank the teaching assistants Ms. Bhanu and Ms. Pratibha for their tutorial sessions on using VMD and GROMACS.

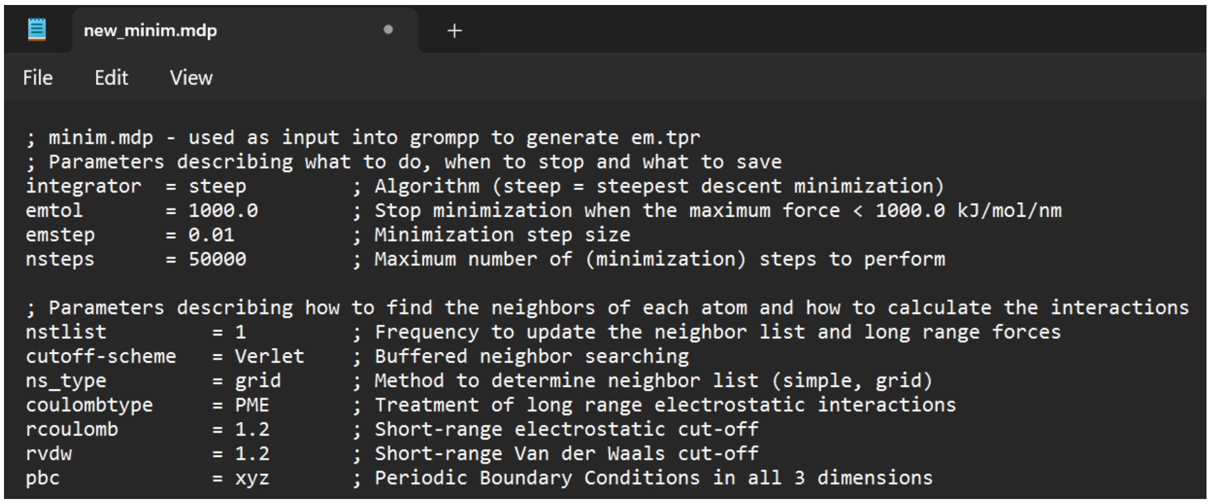
Appendix

`new_ions.mdp`



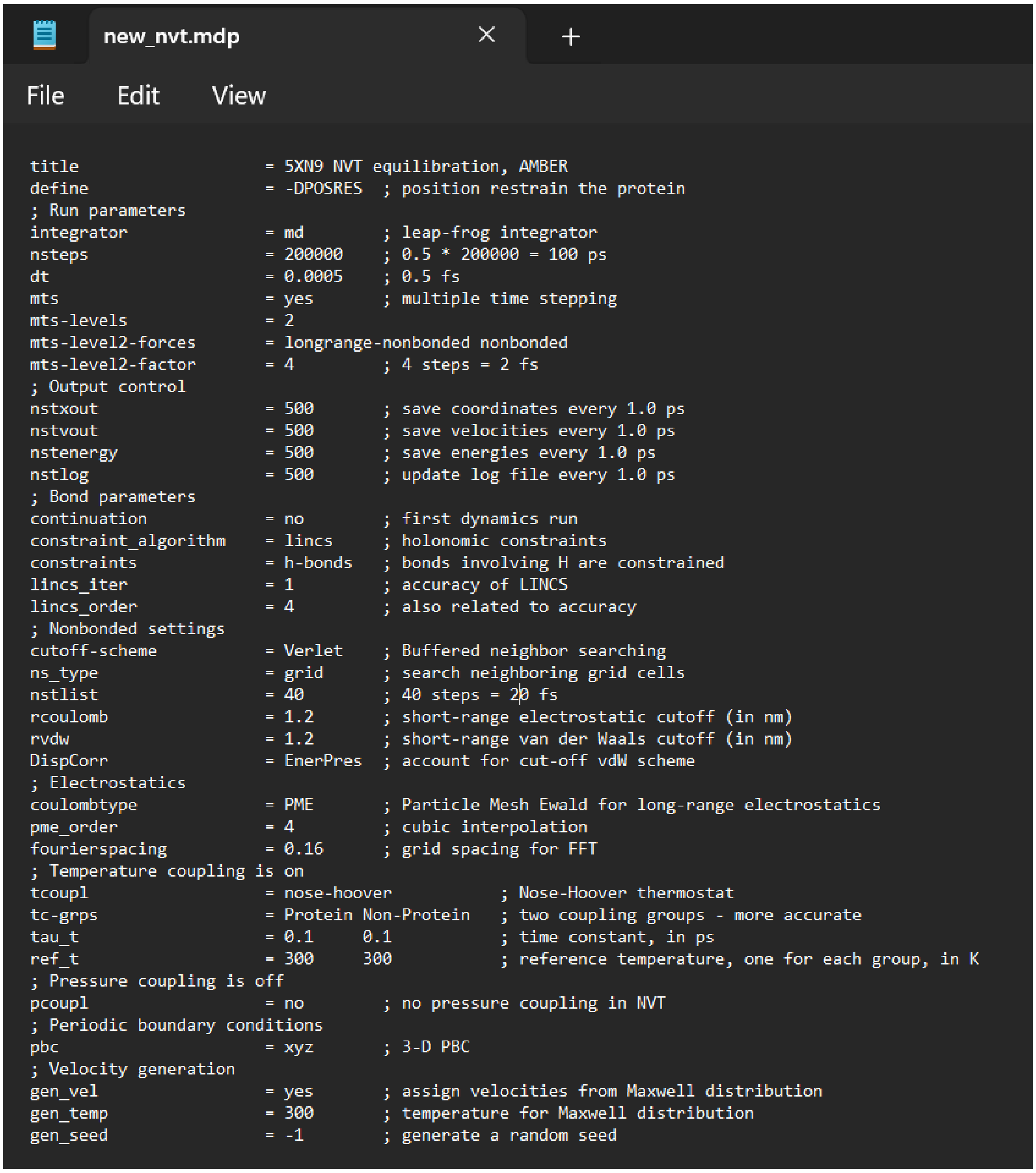
```
; ions.mdp - used as input into grompp to generate ions.tpr
; Parameters describing what to do, when to stop and what to save
integrator  = steep         ; Algorithm (steep = steepest descent minimization)
emtol       = 1000.0        ; Stop minimization when the maximum force < 1000.0 kJ/mol/nm
emstep      = 0.01          ; Minimization step size
nsteps      = 50000         ; Maximum number of (minimization) steps to perform

; Parameters describing how to find the neighbors of each atom and how to calculate the interactions
nstlist         = 1         ; Frequency to update the neighbor list and long range forces
cutoff-scheme   = Verlet    ; Buffered neighbor searching
ns_type         = grid      ; Method to determine neighbor list (simple, grid)
coulombtype     = cutoff    ; Treatment of long range electrostatic interactions. PME can be used when there is no net charge.
rcoulomb        = 1.2       ; Short-range electrostatic cut-off
rvdw            = 1.2       ; Short-range Van der Waals cut-off
pbc             = xyz       ; Periodic Boundary Conditions in all 3 dimensions
```

`new_minim.mdp`



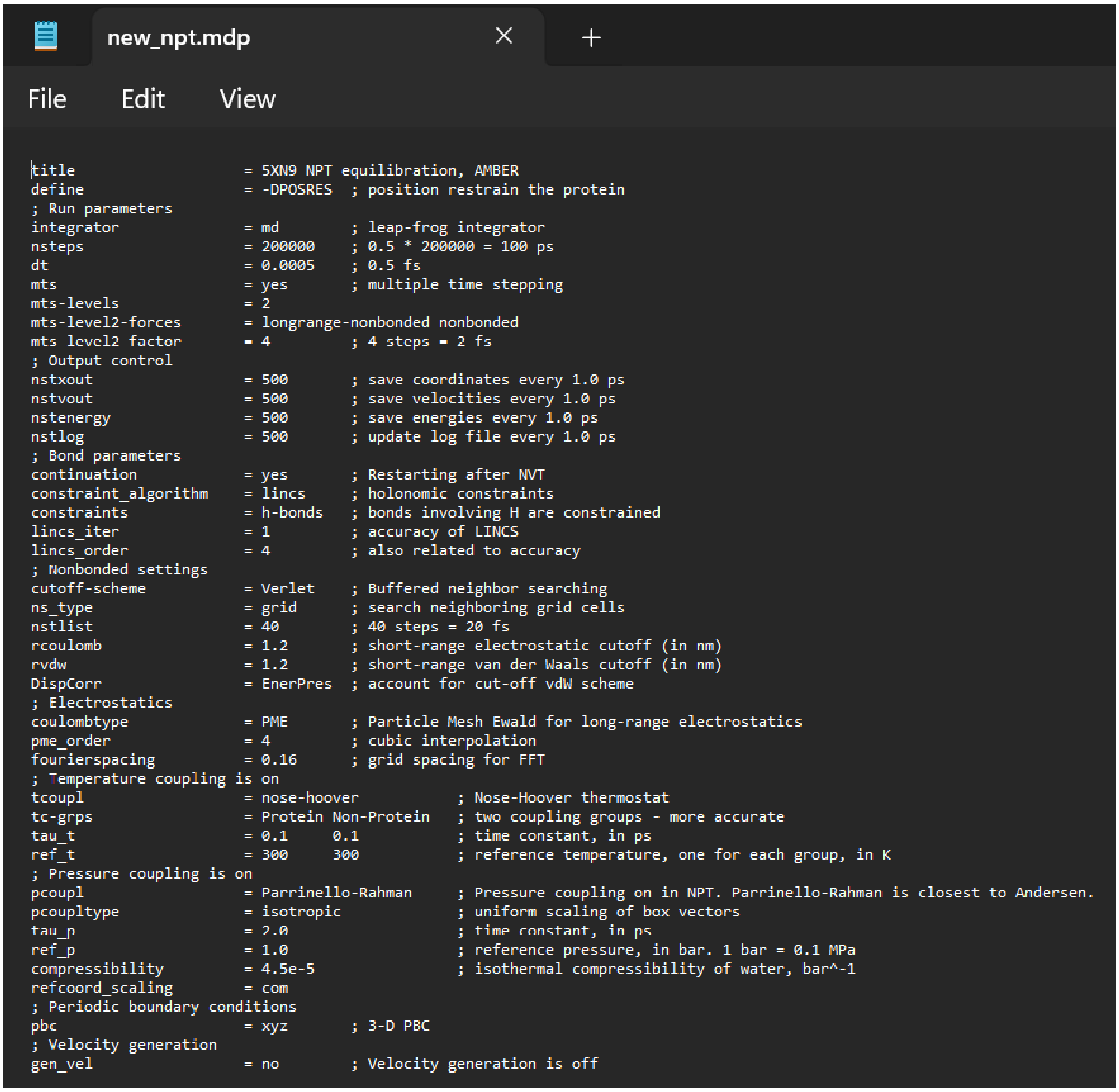
```
; minim.mdp - used as input into grompp to generate em.tpr
; Parameters describing what to do, when to stop and what to save
integrator  = steep         ; Algorithm (steep = steepest descent minimization)
emtol       = 1000.0        ; Stop minimization when the maximum force < 1000.0 kJ/mol/nm
emstep      = 0.01          ; Minimization step size
nsteps      = 50000         ; Maximum number of (minimization) steps to perform

; Parameters describing how to find the neighbors of each atom and how to calculate the interactions
nstlist         = 1         ; Frequency to update the neighbor list and long range forces
cutoff-scheme   = Verlet    ; Buffered neighbor searching
ns_type         = grid      ; Method to determine neighbor list (simple, grid)
coulombtype     = PME       ; Treatment of long range electrostatic interactions
rcoulomb        = 1.2       ; Short-range electrostatic cut-off
rvdw            = 1.2       ; Short-range Van der Waals cut-off
pbc             = xyz       ; Periodic Boundary Conditions in all 3 dimensions
```

new_nvt.mdp



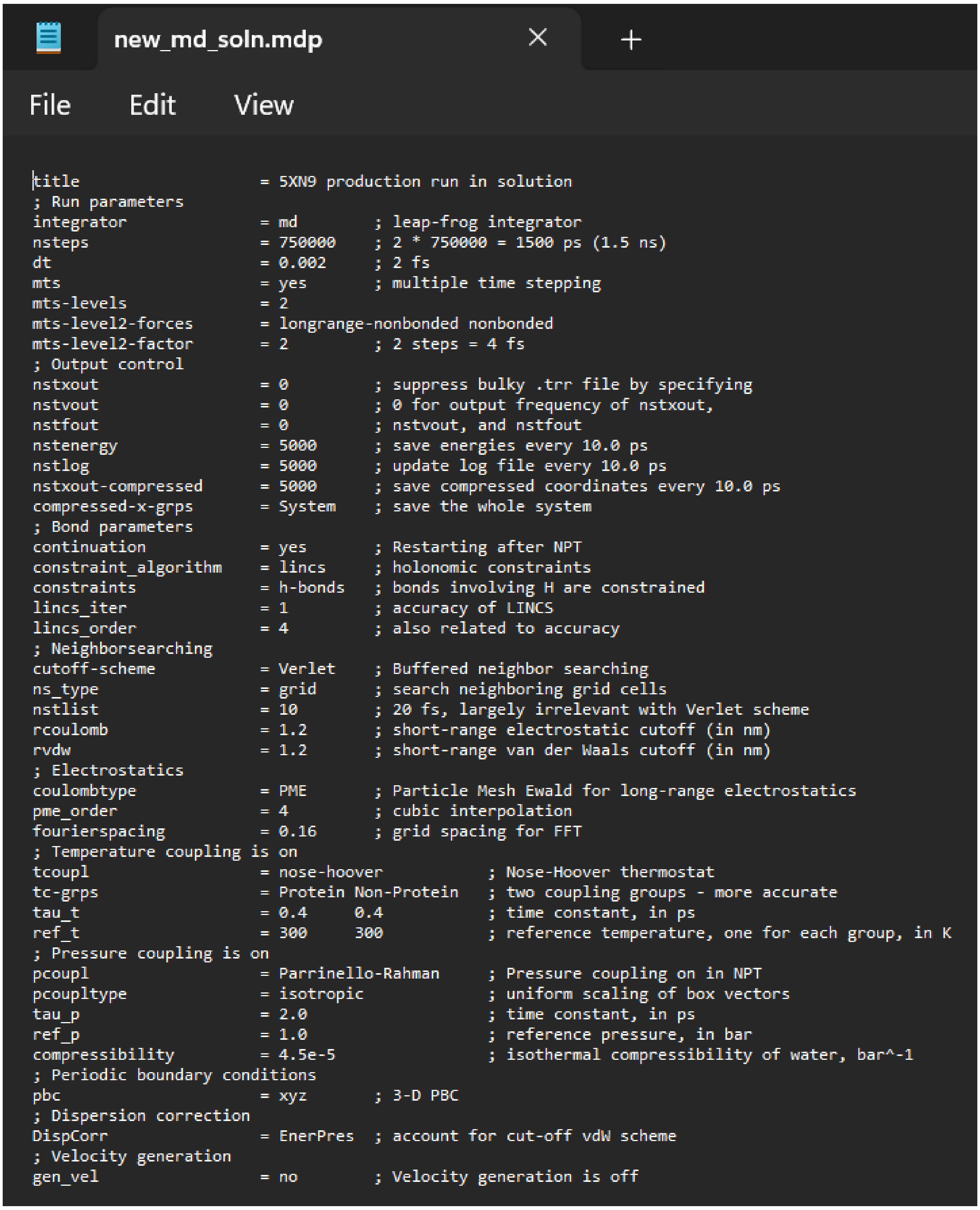
```
title                    = 5XN9 NVT equilibration, AMBER
define                   = -DPOSRES  ; position restrain the protein
; Run parameters
integrator               = md        ; leap-frog integrator
nsteps                   = 200000    ; 0.5 * 200000 = 100 ps
dt                       = 0.0005    ; 0.5 fs
mts                      = yes       ; multiple time stepping
mts-levels               = 2
mts-level2-forces        = longrange-nonbonded nonbonded
mts-level2-factor        = 4         ; 4 steps = 2 fs
; Output control
nstxout                  = 500       ; save coordinates every 1.0 ps
nstvout                  = 500       ; save velocities every 1.0 ps
nstenergy                = 500       ; save energies every 1.0 ps
nstlog                   = 500       ; update log file every 1.0 ps
; Bond parameters
continuation             = no        ; first dynamics run
constraint_algorithm     = lincs     ; holonomic constraints
constraints              = h-bonds   ; bonds involving H are constrained
lincs_iter               = 1         ; accuracy of LINCS
lincs_order              = 4         ; also related to accuracy
; Nonbonded settings
cutoff-scheme            = Verlet    ; Buffered neighbor searching
ns_type                  = grid      ; search neighboring grid cells
nstlist                  = 40        ; 40 steps = 20 fs
rcoulomb                 = 1.2       ; short-range electrostatic cutoff (in nm)
rvdw                     = 1.2       ; short-range van der Waals cutoff (in nm)
DispCorr                 = EnerPres  ; account for cut-off vdW scheme
; Electrostatics
coulombtype              = PME       ; Particle Mesh Ewald for long-range electrostatics
pme_order                = 4         ; cubic interpolation
fourierspacing           = 0.16      ; grid spacing for FFT
; Temperature coupling is on
tcoupl                   = nose-hoover           ; Nose-Hoover thermostat
tc-grps                  = Protein Non-Protein   ; two coupling groups - more accurate
tau_t                    = 0.1     0.1           ; time constant, in ps
ref_t                    = 300     300           ; reference temperature, one for each group, in K
; Pressure coupling is off
pcoupl                   = no        ; no pressure coupling in NVT
; Periodic boundary conditions
pbc                      = xyz       ; 3-D PBC
; Velocity generation
gen_vel                  = yes       ; assign velocities from Maxwell distribution
gen_temp                 = 300       ; temperature for Maxwell distribution
gen_seed                 = -1        ; generate a random seed
```

new_npt.mdp



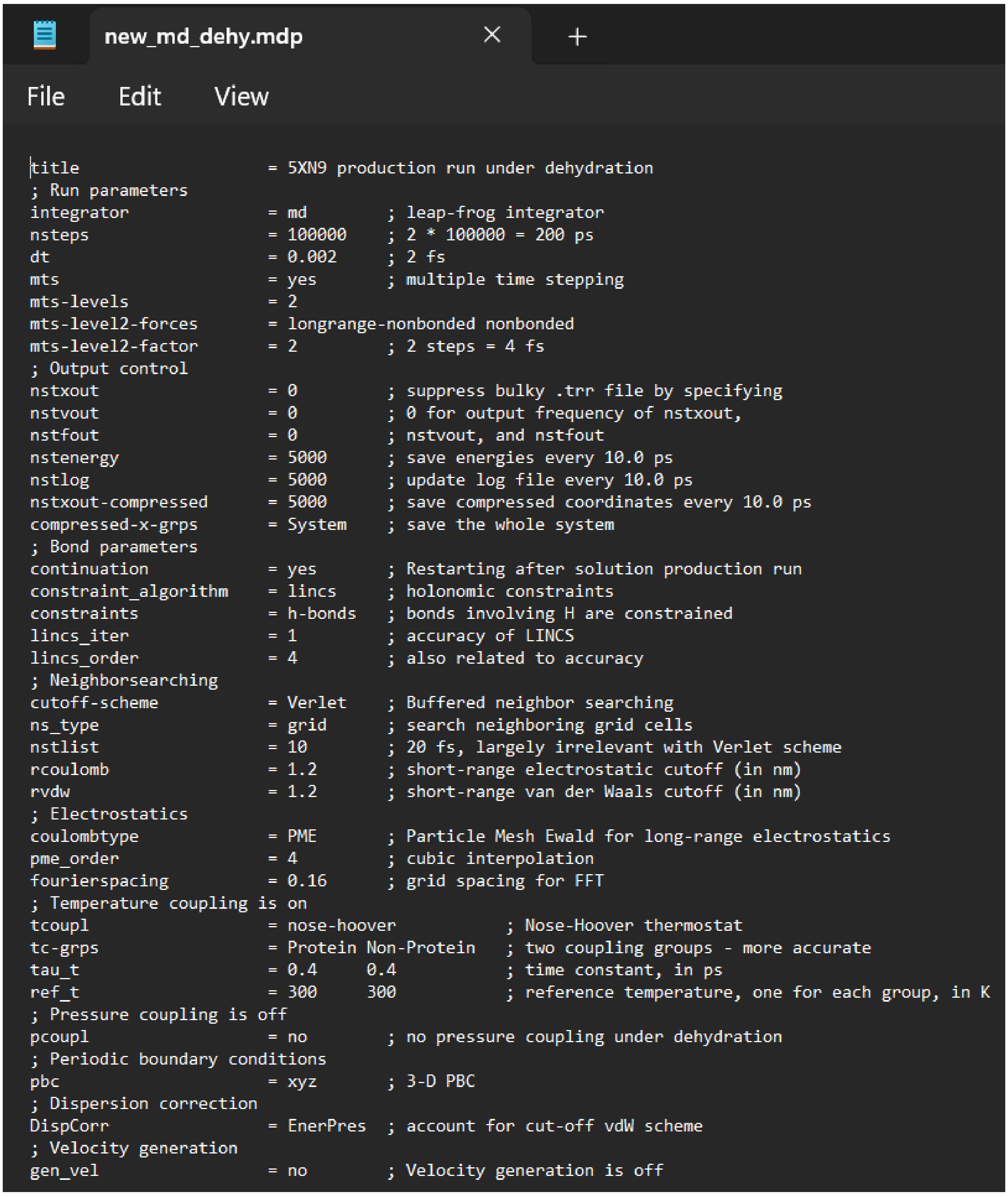
```
title                   = 5XN9 NPT equilibration, AMBER
define                  = -DPOSRES  ; position restrain the protein
; Run parameters
integrator              = md        ; leap-frog integrator
nsteps                  = 200000    ; 0.5 * 200000 = 100 ps
dt                      = 0.0005    ; 0.5 fs
mts                     = yes       ; multiple time stepping
mts-levels              = 2
mts-level2-forces       = longrange-nonbonded nonbonded
mts-level2-factor       = 4         ; 4 steps = 2 fs
; Output control
nstxout                 = 500       ; save coordinates every 1.0 ps
nstvout                 = 500       ; save velocities every 1.0 ps
nstenergy               = 500       ; save energies every 1.0 ps
nstlog                  = 500       ; update log file every 1.0 ps
; Bond parameters
continuation            = yes       ; Restarting after NVT
constraint_algorithm    = lincs     ; holonomic constraints
constraints             = h-bonds   ; bonds involving H are constrained
lincs_iter              = 1         ; accuracy of LINCS
lincs_order             = 4         ; also related to accuracy
; Nonbonded settings
cutoff-scheme           = Verlet    ; Buffered neighbor searching
ns_type                 = grid      ; search neighboring grid cells
nstlist                 = 40        ; 40 steps = 20 fs
rcoulomb                = 1.2       ; short-range electrostatic cutoff (in nm)
rvdw                    = 1.2       ; short-range van der Waals cutoff (in nm)
DispCorr                = EnerPres  ; account for cut-off vdW scheme
; Electrostatics
coulombtype             = PME       ; Particle Mesh Ewald for long-range electrostatics
pme_order               = 4         ; cubic interpolation
fourierspacing          = 0.16      ; grid spacing for FFT
; Temperature coupling is on
tcoupl                  = nose-hoover           ; Nose-Hoover thermostat
tc-grps                 = Protein Non-Protein   ; two coupling groups - more accurate
tau_t                   = 0.1     0.1           ; time constant, in ps
ref_t                   = 300     300           ; reference temperature, one for each group, in K
; Pressure coupling is on
pcoupl                  = Parrinello-Rahman     ; Pressure coupling on in NPT. Parrinello-Rahman is closest to Andersen.
pcoupltype              = isotropic             ; uniform scaling of box vectors
tau_p                   = 2.0                   ; time constant, in ps
ref_p                   = 1.0                   ; reference pressure, in bar. 1 bar = 0.1 MPa
compressibility         = 4.5e-5                ; isothermal compressibility of water, bar^-1
refcoord_scaling        = com
; Periodic boundary conditions
pbc                     = xyz       ; 3-D PBC
; Velocity generation
gen_vel                 = no        ; Velocity generation is off
```

new_md_soln.mdp



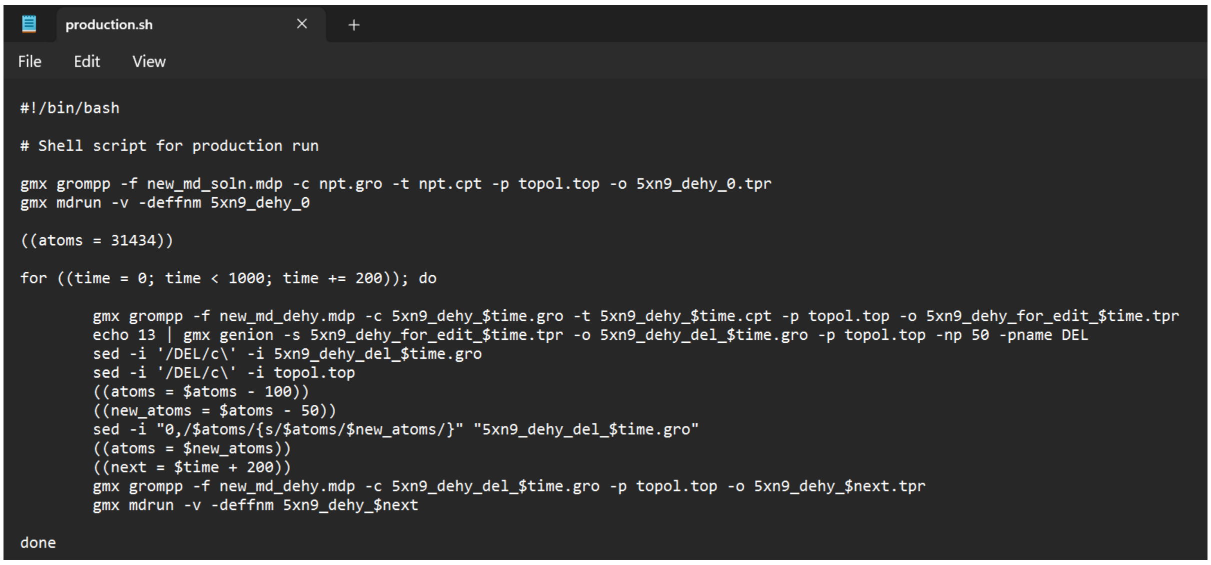
```
title                   = 5XN9 production run in solution
; Run parameters
integrator              = md        ; leap-frog integrator
nsteps                  = 750000    ; 2 * 750000 = 1500 ps (1.5 ns)
dt                      = 0.002     ; 2 fs
mts                     = yes       ; multiple time stepping
mts-levels              = 2
mts-level2-forces       = longrange-nonbonded nonbonded
mts-level2-factor       = 2         ; 2 steps = 4 fs
; Output control
nstxout                 = 0         ; suppress bulky .trr file by specifying
nstvout                 = 0         ; 0 for output frequency of nstxout,
nstfout                 = 0         ; nstvout, and nstfout
nstenergy               = 5000      ; save energies every 10.0 ps
nstlog                  = 5000      ; update log file every 10.0 ps
nstxout-compressed      = 5000      ; save compressed coordinates every 10.0 ps
compressed-x-grps       = System    ; save the whole system
; Bond parameters
continuation            = yes       ; Restarting after NPT
constraint_algorithm    = lincs     ; holonomic constraints
constraints             = h-bonds   ; bonds involving H are constrained
lincs_iter              = 1         ; accuracy of LINCS
lincs_order             = 4         ; also related to accuracy
; Neighborsearching
cutoff-scheme           = Verlet    ; Buffered neighbor searching
ns_type                 = grid      ; search neighboring grid cells
nstlist                 = 10        ; 20 fs, largely irrelevant with Verlet scheme
rcoulomb                = 1.2       ; short-range electrostatic cutoff (in nm)
rvdw                    = 1.2       ; short-range van der Waals cutoff (in nm)
; Electrostatics
coulombtype             = PME       ; Particle Mesh Ewald for long-range electrostatics
pme_order               = 4         ; cubic interpolation
fourierspacing          = 0.16      ; grid spacing for FFT
; Temperature coupling is on
tcoupl                  = nose-hoover           ; Nose-Hoover thermostat
tc-grps                 = Protein Non-Protein   ; two coupling groups - more accurate
tau_t                   = 0.4     0.4           ; time constant, in ps
ref_t                   = 300     300           ; reference temperature, one for each group, in K
; Pressure coupling is on
pcoupl                  = Parrinello-Rahman     ; Pressure coupling on in NPT
pcoupltype              = isotropic             ; uniform scaling of box vectors
tau_p                   = 2.0                   ; time constant, in ps
ref_p                   = 1.0                   ; reference pressure, in bar
compressibility         = 4.5e-5                ; isothermal compressibility of water, bar^-1
; Periodic boundary conditions
pbc                     = xyz       ; 3-D PBC
; Dispersion correction
DispCorr                = EnerPres  ; account for cut-off vdW scheme
; Velocity generation
gen_vel                 = no        ; Velocity generation is off
```

new_md_dehy.mdp



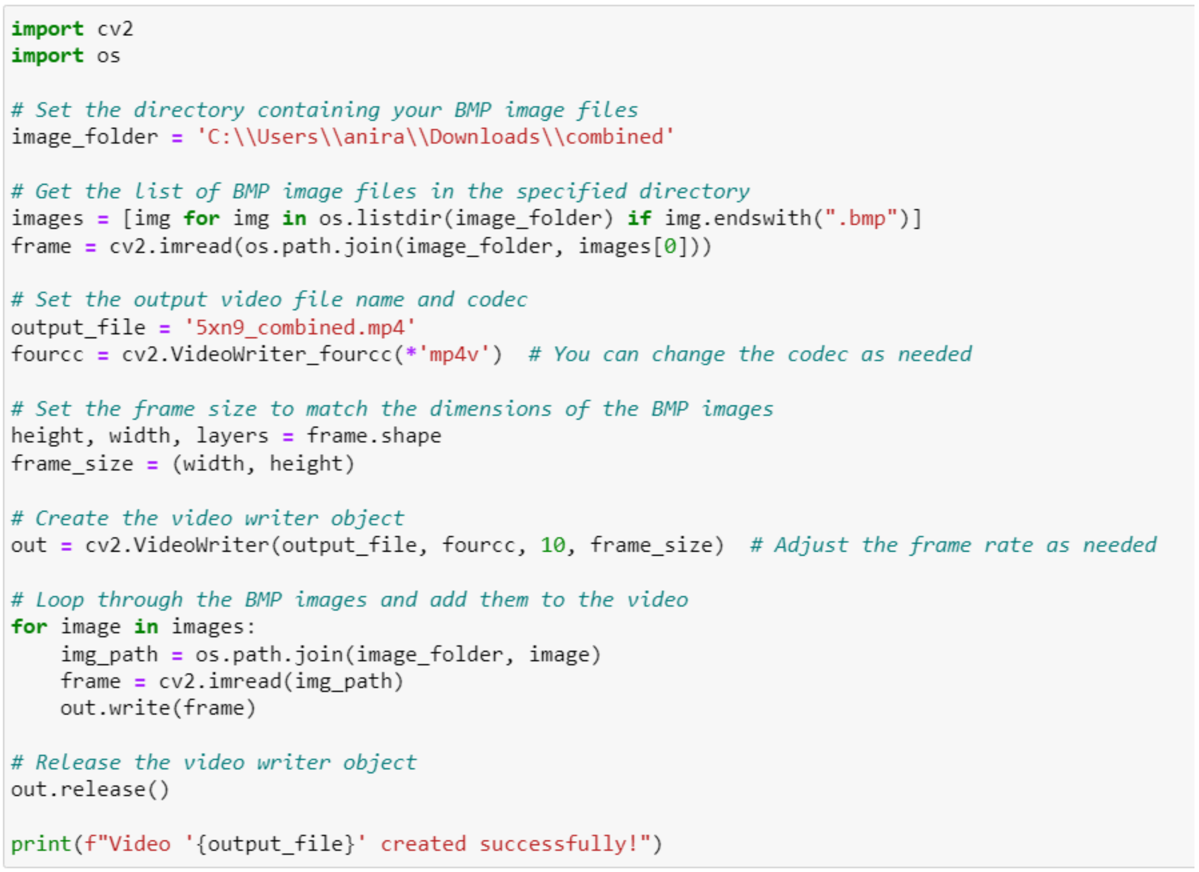
```
title                    = 5XN9 production run under dehydration
; Run parameters
integrator               = md        ; leap-frog integrator
nsteps                   = 100000    ; 2 * 100000 = 200 ps
dt                       = 0.002     ; 2 fs
mts                      = yes       ; multiple time stepping
mts-levels               = 2
mts-level2-forces        = longrange-nonbonded nonbonded
mts-level2-factor        = 2         ; 2 steps = 4 fs
; Output control
nstxout                  = 0         ; suppress bulky .trr file by specifying
nstvout                  = 0         ; 0 for output frequency of nstxout,
nstfout                  = 0         ; nstvout, and nstfout
nstenergy                = 5000      ; save energies every 10.0 ps
nstlog                   = 5000      ; update log file every 10.0 ps
nstxout-compressed       = 5000      ; save compressed coordinates every 10.0 ps
compressed-x-grps        = System    ; save the whole system
; Bond parameters
continuation             = yes       ; Restarting after solution production run
constraint_algorithm     = lincs     ; holonomic constraints
constraints              = h-bonds   ; bonds involving H are constrained
lincs_iter               = 1         ; accuracy of LINCS
lincs_order              = 4         ; also related to accuracy
; Neighborsearching
cutoff-scheme            = Verlet    ; Buffered neighbor searching
ns_type                  = grid      ; search neighboring grid cells
nstlist                  = 10        ; 20 fs, largely irrelevant with Verlet scheme
rcoulomb                 = 1.2       ; short-range electrostatic cutoff (in nm)
rvdw                     = 1.2       ; short-range van der Waals cutoff (in nm)
; Electrostatics
coulombtype              = PME       ; Particle Mesh Ewald for long-range electrostatics
pme_order                = 4         ; cubic interpolation
fourierspacing           = 0.16      ; grid spacing for FFT
; Temperature coupling is on
tcoupl                   = nose-hoover           ; Nose-Hoover thermostat
tc-grps                  = Protein Non-Protein   ; two coupling groups - more accurate
tau_t                    = 0.4     0.4           ; time constant, in ps
ref_t                    = 300     300           ; reference temperature, one for each group, in K
; Pressure coupling is off
pcoupl                   = no        ; no pressure coupling under dehydration
; Periodic boundary conditions
pbc                      = xyz       ; 3-D PBC
; Dispersion correction
DispCorr                 = EnerPres  ; account for cut-off vdW scheme
; Velocity generation
gen_vel                  = no        ; Velocity generation is off
```

production.sh



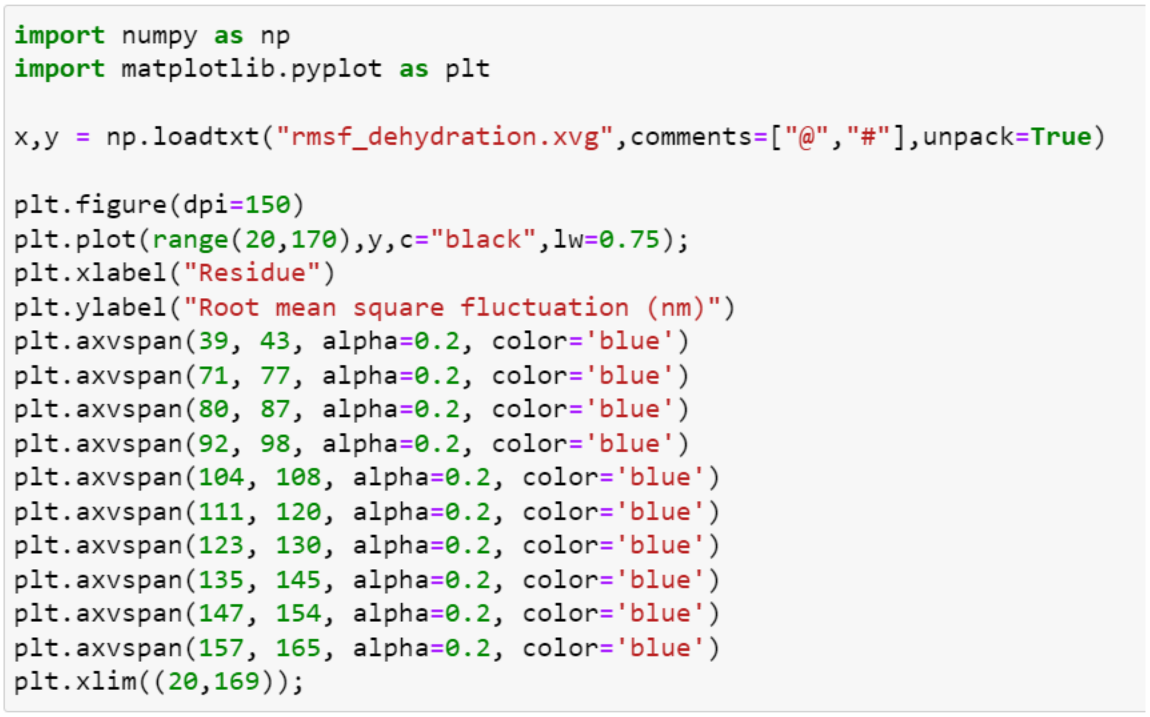
```bash
#!/bin/bash

# Shell script for production run

gmx grompp -f new_md_soln.mdp -c npt.gro -t npt.cpt -p topol.top -o 5xn9_dehy_0.tpr
gmx mdrun -v -deffnm 5xn9_dehy_0

((atoms = 31434))

for ((time = 0; time < 1000; time += 200)); do

        gmx grompp -f new_md_dehy.mdp -c 5xn9_dehy_$time.gro -t 5xn9_dehy_$time.cpt -p topol.top -o 5xn9_dehy_for_edit_$time.tpr
        echo 13 | gmx genion -s 5xn9_dehy_for_edit_$time.tpr -o 5xn9_dehy_del_$time.gro -p topol.top -np 50 -pname DEL
        sed -i '/DEL/c\' -i 5xn9_dehy_del_$time.gro
        sed -i '/DEL/c\' -i topol.top
        ((atoms = $atoms - 100))
        ((new_atoms = $atoms - 50))
        sed -i "0,/$atoms/{s/$atoms/$new_atoms/}" "5xn9_dehy_del_$time.gro"
        ((atoms = $new_atoms))
        ((next = $time + 200))
        gmx grompp -f new_md_dehy.mdp -c 5xn9_dehy_del_$time.gro -p topol.top -o 5xn9_dehy_$next.tpr
        gmx mdrun -v -deffnm 5xn9_dehy_$next

done
```

Python code for making the movies



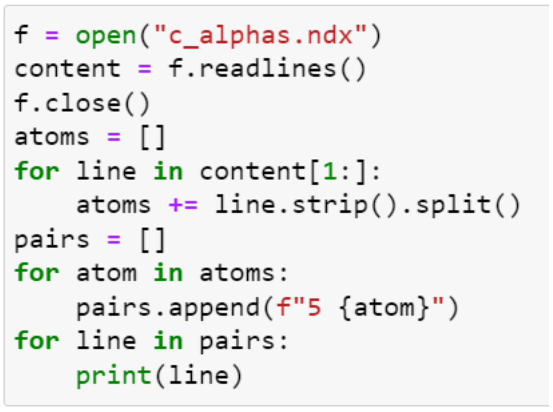
```python
import cv2
import os

# Set the directory containing your BMP image files
image_folder = 'C:\\Users\\anira\\Downloads\\combined'

# Get the list of BMP image files in the specified directory
images = [img for img in os.listdir(image_folder) if img.endswith(".bmp")]
frame = cv2.imread(os.path.join(image_folder, images[0]))

# Set the output video file name and codec
output_file = '5xn9_combined.mp4'
fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # You can change the codec as needed

# Set the frame size to match the dimensions of the BMP images
height, width, layers = frame.shape
frame_size = (width, height)

# Create the video writer object
out = cv2.VideoWriter(output_file, fourcc, 10, frame_size)  # Adjust the frame rate as needed

# Loop through the BMP images and add them to the video
for image in images:
    img_path = os.path.join(image_folder, image)
    frame = cv2.imread(img_path)
    out.write(frame)

# Release the video writer object
out.release()

print(f"Video '{output_file}' created successfully!")
```

Python code for plotting root mean square fluctuation



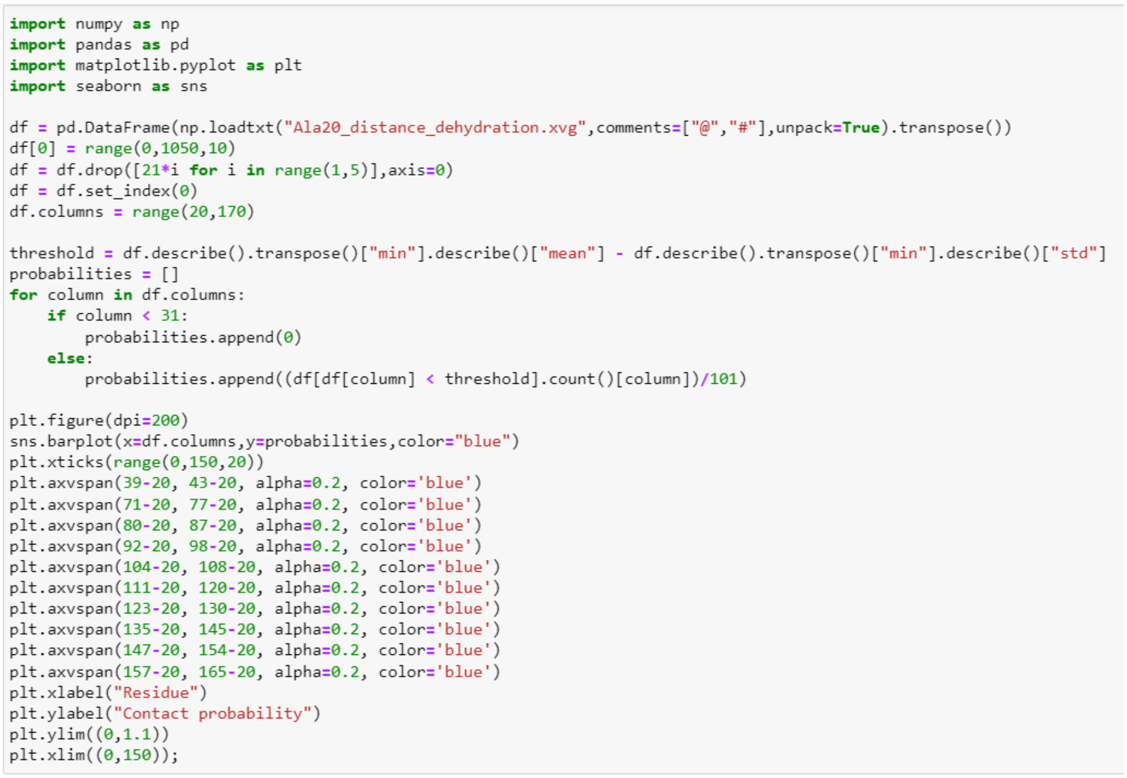
```python
import numpy as np
import matplotlib.pyplot as plt

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

plt.figure(dpi=150)
plt.plot(range(20,170),y,c="black",lw=0.75);
plt.xlabel("Residue")
plt.ylabel("Root mean square fluctuation (nm)")
plt.axvspan(39, 43, alpha=0.2, color='blue')
plt.axvspan(71, 77, alpha=0.2, color='blue')
plt.axvspan(80, 87, alpha=0.2, color='blue')
plt.axvspan(92, 98, alpha=0.2, color='blue')
plt.axvspan(104, 108, alpha=0.2, color='blue')
plt.axvspan(111, 120, alpha=0.2, color='blue')
plt.axvspan(123, 130, alpha=0.2, color='blue')
plt.axvspan(135, 145, alpha=0.2, color='blue')
plt.axvspan(147, 154, alpha=0.2, color='blue')
plt.axvspan(157, 165, alpha=0.2, color='blue')
plt.xlim((20,169));
```

Python code for generating pairwise atom numbers



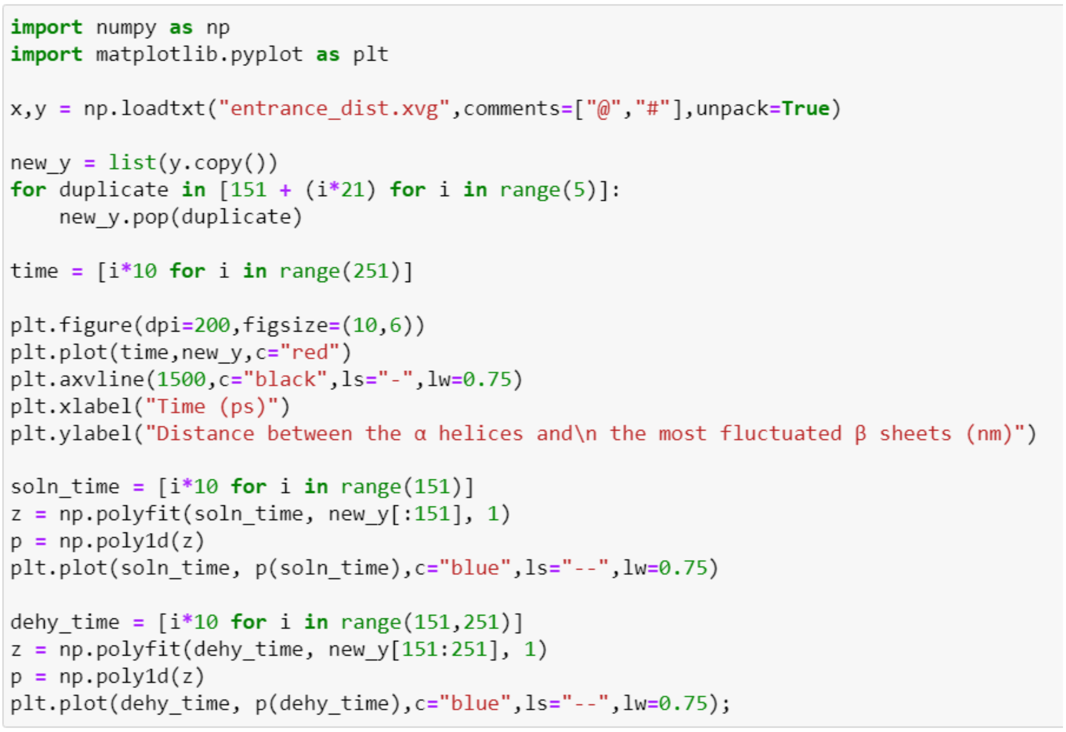
```python
f = open("c_alphas.ndx")
content = f.readlines()
f.close()
atoms = []
for line in content[1:]:
    atoms += line.strip().split()
pairs = []
for atom in atoms:
    pairs.append(f"5 {atom}")
for line in pairs:
    print(line)
```

Python code for Ala20 distance analysis and plotting



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.DataFrame(np.loadtxt("Ala20_distance_dehydration.xvg",comments=["@","#"],unpack=True).transpose())
df[0] = range(0,1050,10)
df = df.drop([21*i for i in range(1,5)],axis=0)
df = df.set_index(0)
df.columns = range(20,170)

threshold = df.describe().transpose()["min"].describe()["mean"] - df.describe().transpose()["min"].describe()["std"]
probabilities = []
for column in df.columns:
    if column < 31:
        probabilities.append(0)
    else:
        probabilities.append((df[df[column] < threshold].count()[column])/101)

plt.figure(dpi=200)
sns.barplot(x=df.columns,y=probabilities,color="blue")
plt.xticks(range(0,150,20))
plt.axvspan(39-20, 43-20, alpha=0.2, color='blue')
plt.axvspan(71-20, 77-20, alpha=0.2, color='blue')
plt.axvspan(80-20, 87-20, alpha=0.2, color='blue')
plt.axvspan(92-20, 98-20, alpha=0.2, color='blue')
plt.axvspan(104-20, 108-20, alpha=0.2, color='blue')
plt.axvspan(111-20, 120-20, alpha=0.2, color='blue')
plt.axvspan(123-20, 130-20, alpha=0.2, color='blue')
plt.axvspan(135-20, 145-20, alpha=0.2, color='blue')
plt.axvspan(147-20, 154-20, alpha=0.2, color='blue')
plt.axvspan(157-20, 165-20, alpha=0.2, color='blue')
plt.xlabel("Residue")
plt.ylabel("Contact probability")
plt.ylim((0,1.1))
plt.xlim((0,150));
```

Python code for plotting entrance region distance

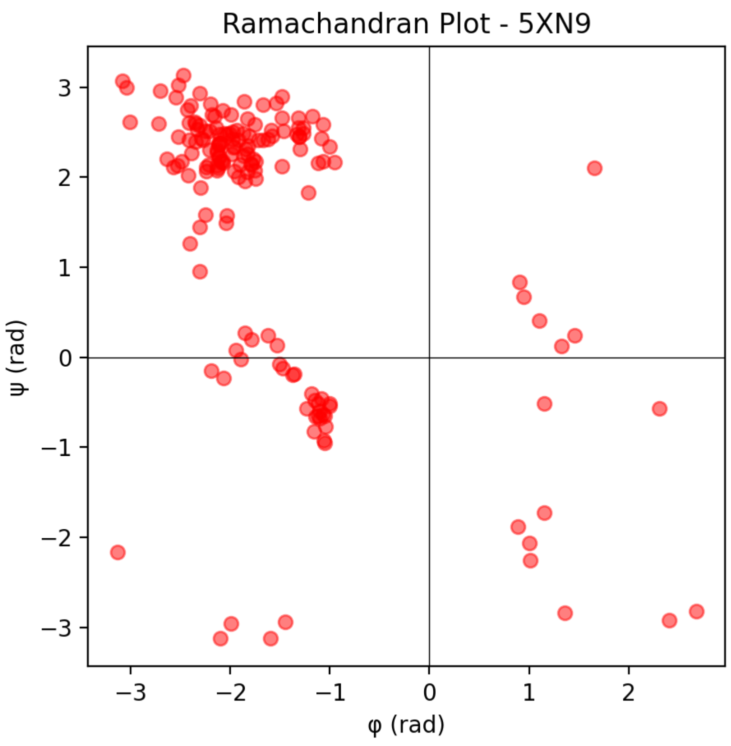


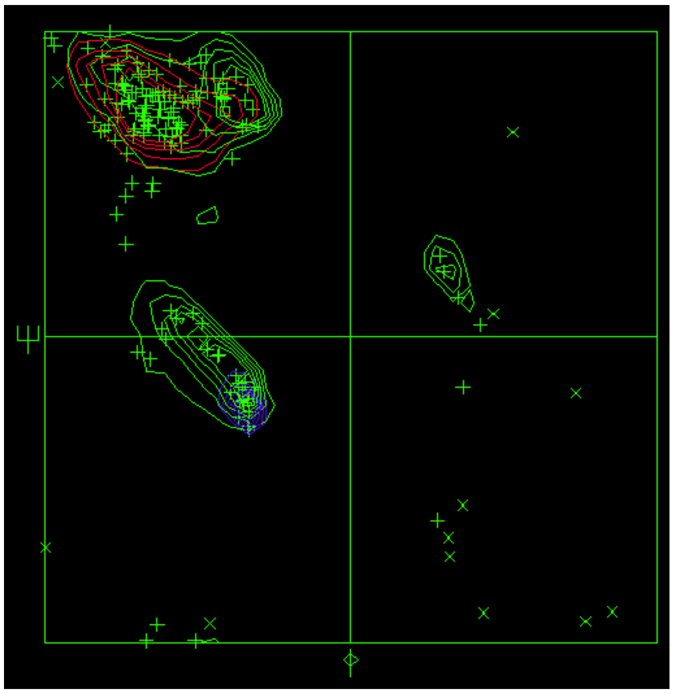
```python
import numpy as np
import matplotlib.pyplot as plt

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

new_y = list(y.copy())
for duplicate in [151 + (i*21) for i in range(5)]:
    new_y.pop(duplicate)

time = [i*10 for i in range(251)]

plt.figure(dpi=200,figsize=(10,6))
plt.plot(time,new_y,c="red")
plt.axvline(1500,c="black",ls="-",lw=0.75)
plt.xlabel("Time (ps)")
plt.ylabel("Distance between the α helices and\n the most fluctuated β sheets (nm)")

soln_time = [i*10 for i in range(151)]
z = np.polyfit(soln_time, new_y[:151], 1)
p = np.poly1d(z)
plt.plot(soln_time, p(soln_time),c="blue",ls="--",lw=0.75)

dehy_time = [i*10 for i in range(151,251)]
z = np.polyfit(dehy_time, new_y[151:251], 1)
p = np.poly1d(z)
plt.plot(dehy_time, p(dehy_time),c="blue",ls="--",lw=0.75);
```

Ramachandran Plot

A Python code was written to compute the Ramachandran plot for the protein being simulated (`5xn9_completed.pdb`). The result obtained was compared to that produced by the WHAT IF web server.

The Python code can be found in the Google Drive link shared earlier.





The plot on the left is generated using the Python code. The plot on the right is generated by the WHAT IF server. The two plots are identical.