Biotechnology and Biological Sciences Research Council

University of Nottingham
UK | CHINA | MALAYSIA

Molecular Dynamics Workshop

Ross Amory
pcyra2@nottingham.ac.uk

School of Chemistry, University of Nottingham
Wednesday 31st May, 2023

Contents

1 How to obtain software

Here we will go through the steps to install all required software for this workshop. It is presumed that you are working either within a standard Linux terminal or using Windows subsystem for linux 2 (WSL2) from within Windows. If you are using any other set-up, you will have to find out how to install the required software separately.

1.a Anaconda

Install python and anaconda using the following commands. You should do this within a WSL2 terminal if you are not natively working within linux.

Command 1.1: Commands to install anaconda and python.

bash command

```
$ wget https://repo.anaconda.com/archive/Anaconda3-2023.03-1-Linux-
  ↪ x86_64.sh
$ sh Anaconda3-2023.03-1-Linux-x86_64.sh
$ # Accept the license agreement and use all defaults suggested.
```

1.b AMBER

Install AmberTools22 using anaconda as this is the simplest way and suitable for this tutorial.

Command 1.2: Commands to install AmberTools22

bash command

```
$ conda create --name AmberTools22
$     y   # proceed with creating the AmberTools22 conda environment
$ conda activate AmberTools22
$ conda install -c conda-forge ambertools=22 compilers
$     y   # proceed with installing
```

1.c Acpype

Install acpype using anaconda. In theory, this will also install ambertools, however, I find that having a separate install for both is best as you have greater control when just using amber.

Command 1.3: Commands to install Acpype

bash command

```
$ conda create --name acpype
$     y   # proceed with creating the environment
$ conda activate acpype
$ conda install -c conda-forge acpype
$     y   # proceed with installing
```

1.d gnina

Below is the quick way of installing gnina. It simply downloads a pre-compiled binary of the software for an individual to use, however it is strongly recommended that you download the source code and compile it yourself for a more optimised experience.

Command 1.4: Commands to install gnina.

bash command

```
$ wget ↩
  ↪ https://github.com/gnina/gnina/releases/download/v1.0.3/gnina
$ chmod +x gnina
$ # It is then recommended to add this to your PATH, .bashrc or ↩
  ↪ modules in order to easily run this later in the workshop.
```

2 Theory

In this section, a brief introduction to the theory behind the activities performed in the workshop will be given. This is not a comprehensive theory section and further reading is strongly recommended to fully understand the methods.

2.a Molecular Dynamics

Due to the size of biological systems, current computational power is unable to simulate every atom within an enzyme at a true quantium mechanics (QM) level and so some approximations must be made in order to simplify the system. The traditional method for simulating large biological molecules such as enzymes is to use a classical physics (molecular mechanics (MM)) approach (Equation: 2.3), which includes using empirical force fields derived from classical physics in order to describe the system. The use of classical physics allows for much simpler calculations and so increases computational efficiency dramatically, allowing us to simulate millions of atoms in a relatively short time scale.

$$\begin{aligned} E_{Total} &= E_{Bonded} + E_{Non-Bonded} \\ E_{Bonded} &= E_{Bond} + E_{Angle} + E_{Dihedral} \\ E_{Non-Bonded} &= E_{Electrostatic} + E_{van\ der\ Waals} \end{aligned} \tag{2.1}$$

Molecular mechanics requires empirical forcefields(1) to describe the behaviours of different atoms (and more importantly biological residues) to describe the energies of the different interactions within a system and so selecting a suitable forcefield which has been parametrised for your use case is important for achieving accurate results. For enzymes, the two most commonly used forcefields are CHARMM(2) and AMBER(3) which were both parametrised for proteins and nucleic acids.

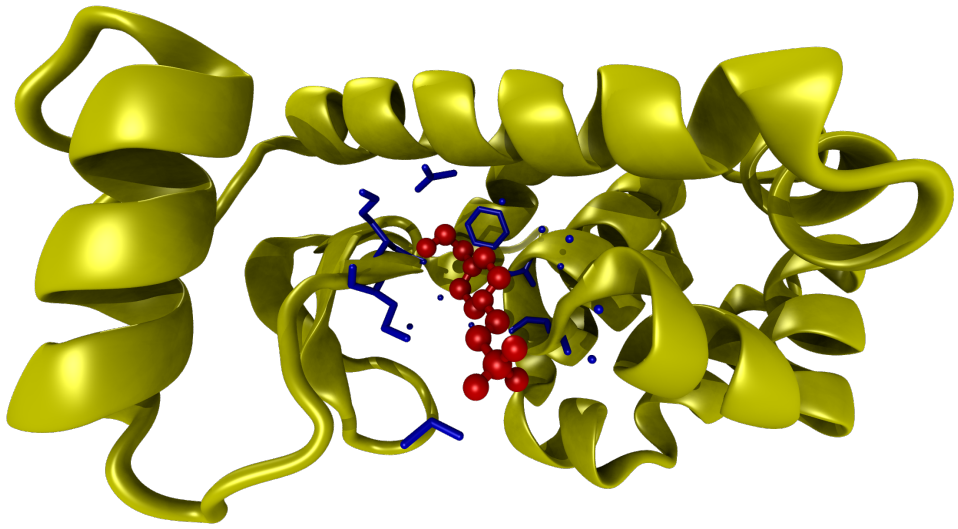

Figure 1: A visual representation of a QM/MM calculation where the atoms in yellow are treated at the MM level of theory, the atoms in red are treated at the QM level and the atoms in blue are treated at both levels of theory.

These MM methods can be combined with QM methods in order to calculate the reaction centre with high accuracy QM theory whilst also including interactions with the wider environment, specifically the electrostatic, dispersion and structural effects, which can control the structure of the reaction site and the thermodynamics of the system. In QM/MM you split the system into two or three sections (usually three) with your reaction site and any atoms which are known to be directly involved being put into the QM region. Then a thin layer around this region is included

as a hybrid zone which allows for communication between the two methods and is used when the zone boundary intersects a covalent chemical bond. These atoms are usually calculated at both levels of theory. Finally, the rest of the system is treated at the simplest level of theory (MM) (Figure: 1).

2.b Docking

Molecular docking is a method of estimating the interactions between two or more chemical species. There are many different types of docking available including covalent and non-covalent molecular docking. In this workshop, we are interested in the non-covalent docking between a protein and a ligand. Like molecular dynamics, docking uses a set of parameters which describe the molecules within a system and uses these to attempt to estimate how well a given orientation and position of a ligand will bind to a protein. This is known as the 'scoring function' of the docking protocol.

Firstly the docking protocol samples translational, rotational and conformational space where the ligand is repositioned within the protein active site. This is often done using a Monte Carlo search in order to obtain a high number of different binding poses for the ligand.

The different poses are then scored with respect to their scoring function in order to estimate the binding affinity or energy of the interaction between the ligand and molecule. This scoring function can be parameterised to include a variety of chemical information including electronegativity, van der Waals (VDW) interactions, hydrogen bonding and solvation effects. Each of the different poses generated by the initial sampling is then scored using this function and the best-scoring poses are saved for further evaluation.

Higher-level docking programs can often take their best scoring poses and attempt to minimise them, often using a simple molecular dynamics forcefield. They then re-score the binding poses in order to get an improved estimate of the binding interactions.

Depending on what docking software you chose, the chemical information that it uses to score the binding affinity can change. It is therefore important to understand what parameters the docking score is based on in order to decide whether the docking software is suitable for your system. It is also known that docking accuracy can be heavily system dependent, so often a benchmark is required in order to assess the validity of any docking results. Docking can also be used to estimate the difference in binding energies between two different ligands, however often this is inaccurate and only suitable if the binding affinity difference is large.

2.c MM/GB(PB)SA

Molecular Mechanics/Poisson–Boltzmann Surface Area (MM/PBSA)(4) or Molecular Mechanics/Generalized Born Surface Area (MM/GBSA) are computational methods that attempt to estimate the free energy of binding between two systems. It uses an implicit solvent model, in combination with a molecular mechanics force field and an understanding of the conservation of energy to estimate Gibb's free energy of binding between a protein and ligand system.

MM/PBSA and MM/GBSA are often used interchangeably and the relative performance of both methods can be system dependant with the difference between the two being from the implicit solvent method used. MM/PBSA uses the Poisson-Boltzman solvent model to describe the solvation energies of each model whereas MM/GBSA uses the generalised born solvent model.

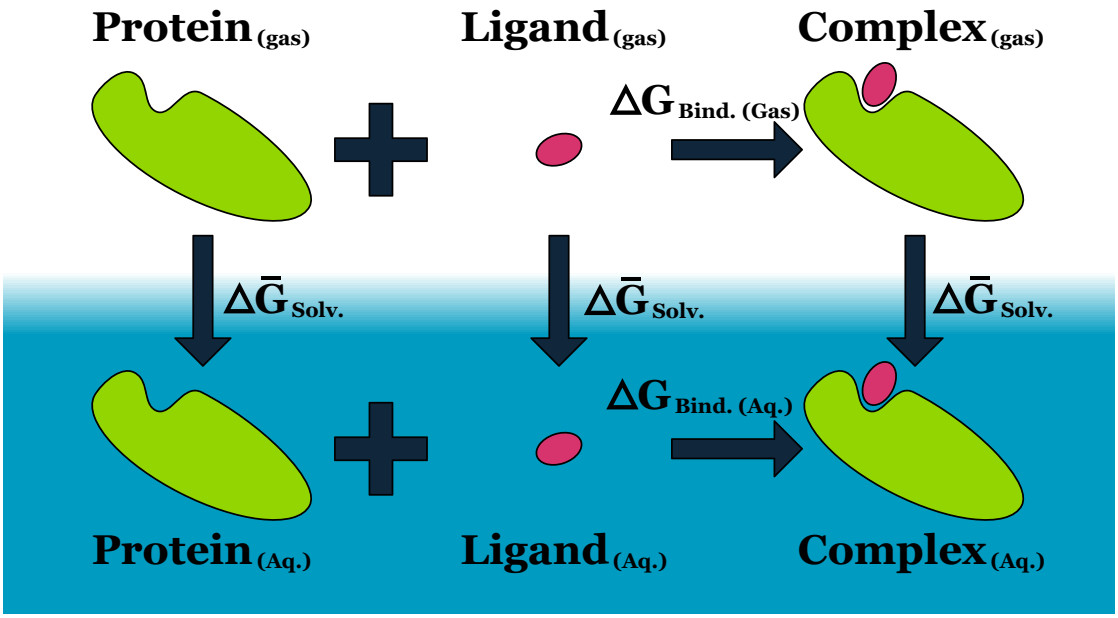

Figure 2: A visual representation of the MM/PB(GB)SA thermodynamic cycle.

The MM/GBSA method requires a traditional molecular dynamics simulation and uses post-processing to calculate the free energy, $\bar{G}$, of the structure through the removal of the explicit waters and counter ions. The free energy of the structure is calculated using equation 2.2.

$$\bar{G} = E_{MM} + G_{PBSA} - TS_{MM} \tag{2.2}$$

Where E_{MM} is the molecular mechanical energy which is taken from the forcefield used within the simulation (equation 2.3).

$$E_{MM} = E_{bond} + E_{angle} + E_{tors} + E_{vdw} + E_{elec} \tag{2.3}$$

G_{PBSA} is the solvation-free energy of the structure, which uses the implicit solvent model in combination with a simple surface area term to estimate the solvation energy. TS_{MM} is the entropy and can be estimated using quasi-harmonic analysis of the trajectory, or normal-mode analysis.

$$\Delta G_{bind} = \Delta \bar{G}_{complex} - \Delta \bar{G}_{prot.} - \Delta \bar{G}_{lig.} \tag{2.4}$$

3 Workshop outline

3.a Goals

The main goal of this workshop is to give you a brief overview of what a standard molecular dynamics (MD) workflow would look like. We shall be covering some of the core techniques used to set up and run a variety of different calculations including **molecular docking**, **molecular dynamics** and **free energy** calculations. We shall also briefly cover some analysis techniques available and some other resources available to further your understanding of the techniques used throughout.

The steps of the workshop are as follows:

- Locate and download the crystal structure
- Dock the ligand into the active site
- Parameterise the ligand
- Create molecular dynamics input files
- Run a simple dynamics simulation
- Perform simple trajectory analysis

- Calculate the binding energy of the ligand

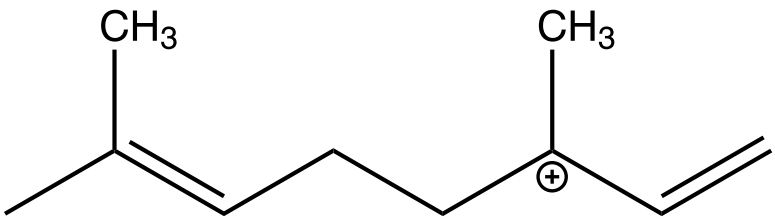

Figure 3: Ligand used within the workshop.

We shall be using a ligand (Figure 3) from a study by Major et. al.(5) due to personal experience working with Terpene synthesis. For more information about the system, please see the previous work.

3.b Software

The key software that we will be using for this workshop is all free to use and should be simple to access and use. Most of the software however requires a basic understanding of how to use the command-line interface (CLI). It is possible to use all of the software from within Windows but Linux is recommended for the smoothest experience. The 'Cheat Codes' in Section 6 at the end of this document has only been tested in a pure Linux operating system (OS).

The required software for this workshop is as follows:

- anaconda
- WSL2 (windows only)
- AMBER(3) [a]
- VMD(6)

[a] It is recommended that you install the full (paid) version of amber for production dynamics due to the GPU support, however for individual machines and the purpose of this workshop, the conda binary distribution is okay.

- gnina(7)
- acpype (8) [a]

Although not required for this workshop, some other software that is available to assist with the MD workflow are as follows:

- CHARMM(9) (MD package)
- gromacs(10) (Open source MD package)
- CGenFF(11) (Forcefield parameterisation tool for the CHARMM forcefield)
- pyMol (graphical user interface (GUI) for molecular visualisation and manipulation)
- AutoDock4(12) (Open source docking software)
- YASARA(13) (GUI for the whole MD workflow)
- Alphafold(14, 15) (Open source A.I. homology modelling software)

4 File types and jargon

There are many different files that are required to run a molecular dynamics simulation. Although many of the files can be used interchangeably, some file types contain extra information which can be useful in specific use cases.

[a] A web server version of acpype is available here for users unable to install, however installing this is highly recommended where possible.

4.a File types

4.a.i Structure

The first group of file types that are crucial, not only to molecular dynamics but to all of computational chemistry are coordinate or structure files. The most common version of these are '.xyz' files which contain the elemental information as well as the 3D spatial coordinates. A simple coordinate file must contain the 3D spatial coordinates of all particles in the system. There are many other pieces of information that can be entered into a structure file such as atomic elements, atom types, chemical charges and bonding information. Open Babel is a tool that has the ability to convert between many different structure file formats and can be incredibly useful when working with a variety of different structure files.

Extension	Extra Information	Human Readable
.xyz	Elements, Number of atoms	Yes
.pdb	Elements, Residue names, Atom types, Protein information	Yes
.mol2	Atom types, Bonds, Charges, Molecule name	Yes
.rst7	Number of atoms, Periodic cell vectors	Yes
.ncrst	Number of atoms, Velocities, Periodic cell vectors	No

Table 2: A non-exhaustive list of structural file types that are used within the molecular dynamics workflow.

4.a.ii Parameter files

Although structure files often contain enough information in order to visualise or run simple calculations on the systems, when a large number of atoms or molecules are included in the system it is often sensible to split these into two files. This is where parameter files become useful as they contain a list of extra information about the particles in the system. This is also useful for dynamical systems where the parameters of the particles do not change but the positional information does. Parameter files are mainly used within molecular dynamics simulations for this reason and it is important to note that they can also be known as topology files within other

simulation packages. The parameter file format that we will be using within the workshop is the '.parm7' file type from Amber.

4.a.iii Trajectory files

Trajectory files in their simplest form are a combination of structure files that allow you to follow the movement of particles within a simulation. As they can contain many different sets of coordinates for a single system, these files can often become large and so they usually contain only the spatial information of the system and so require a parameter file to be understood. There are two types of trajectory files that we will see throughout the workshop; '.nc' and '.binpos'. Both of these files are in binary format in order to reduce the file size and the main reason for using '.binpos' files are that '.nc' files cannot be opened in the Windows version of `VMD`.

4.a.iv Input files

Due to the functionality of most scientific software, it is often useful to create input files that tell the software what type of calculation you are planning to undertake. These are usually plain text files that have specific keywords that are understood by the software.

4.a.v Data files

When running molecular dynamics, you can quickly become overwhelmed by the amount of data that can be outputted. It is therefore often useful to extract specific data into '.dat' files which are usually tabular files that contain plottable data for a specific property. The '.dat' file extension is not exclusive to this usage however and not all of these can be quickly plotted so it is often useful to check what data is contained within such files.

5 Tasks

5.a Initial Tasks

Task 1: Set up the directory structure for the workshop.

Create directories for:

(a) Structure files

(b) Docking simulations

(c) MD simulations

(d) Free energy calculations

Firstly, whenever starting any new project, a working directory needs to be created. This can be done either using the CLI in a terminal using the '`mkdir`' command, or using a GUI file manager. Commands to do this can be found in the cheat codes Command 6.1.

Output 5.1: Folder structure as shown by the `tree` command.

bash output

```
MD_Workshop
├── Structures
├── Docking
├── Dynamics
└── Free_Energy
```

Task 2: Obtaining the structure files.

(a) Obtain the protein crystal structure
  (i) Visit the protein database website
  (ii) Find the protein using code 1n23
  (iii) Download the crystal structure

(b) Save the ligand coordinates files located in the workshop files

(c) Split the complex crystal structure into the protein and ligand

There are a few places where you can find and store crystal structure coordinates, however, the main two that are used are rscb and uniprot. These websites are powerful tools and often group multiple versions of the same protein. Most proteins are published on the rscb database however for unresolved crystal structures, uniprot sometimes also contains the alphafold(*15*) homology model structure.

Although we will not be covering the theory or methods of homology modelling in this workshop, being aware of the technique could potentially be useful when working with an unresolved protein structure. The method allows for an approximate structure to be generated using similar proteins as templates. A homology modelling program with growing popularity is the alphafold(*15*) program which uses artificial intelligence to estimate a crystal structure from a FASTA sequence.

To start this workshop, you will need to visit the rscb website and download the 'pdb' file for the (+)-Bornyl diphosphate synthase which can be found using the code 1N23. If you google the protein code, it usually comes up as one of the first options. If you have any problems with this then use the commands found in the cheat codes Command 6.2. Although there are multiple

versions of this protein, we want the structure that has the diphosphate bound separately to the ligand so that we can dock a terpene ligand into the active site.

You can visualise this protein using `VMD` to check that the protein is correct (Command 5.2). When you open the structure it will likely be in the 'lines' representation, meaning you will struggle to see the protein backbone. It is recommended to change this representation by going into the 'Graphics' menu and selecting 'Representations' (Figure 4). From the new window created, then change the drawing method from 'Lines' to 'NewCartoon' to see the protein backbone structure (Figure 5).

Command 5.2: How to open the structure in VMD

bash command

```
$ vmd 1N23.pdb
```

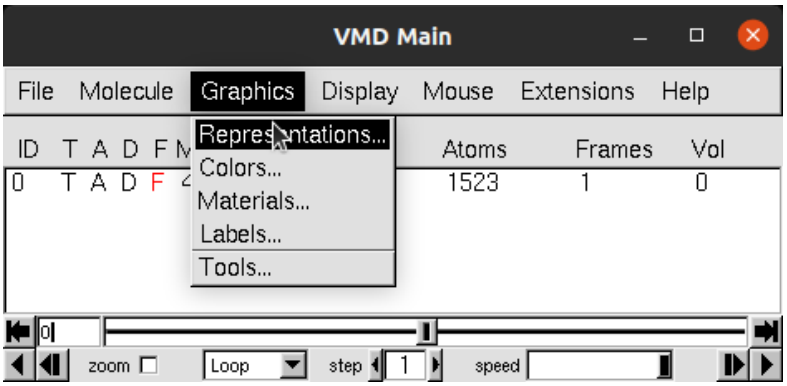

Figure 4: How to access the representations panel in VMD.

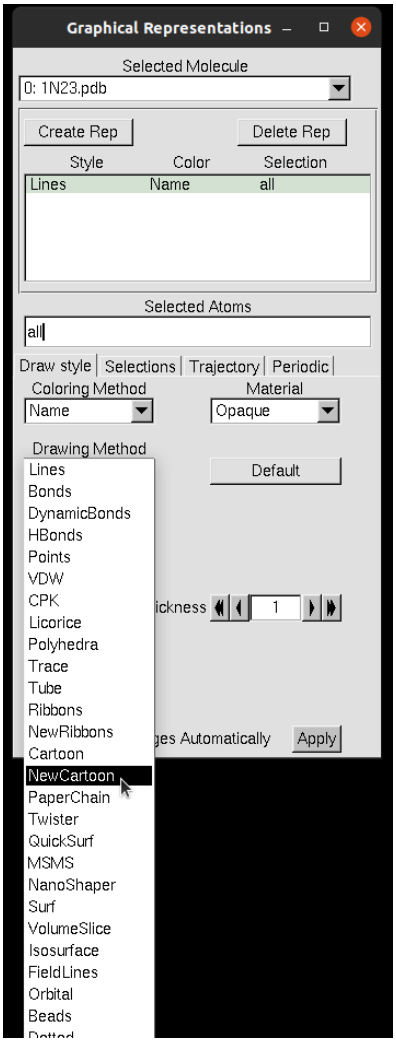

Figure 5: How to change the drawing method in VMD.

Once the protein is correctly visualised, it is often useful to also be able to see the ligand in a 'ball and stick' like representation (known as 'CPK' in VMD). To do this, you first need to know the residue name for the ligand (2BN in this case). You then go back into the graphical representations window and add a new representation using the 'Create Rep' button (Figure 6). From there you change the selected atoms from 'all' to 'resname 2BN' (Figure 7). Finally, you can

then change the drawing method to 'CPK' to see the ligand as a ball and stick (Figure 8). The end result should look like Figure 9.

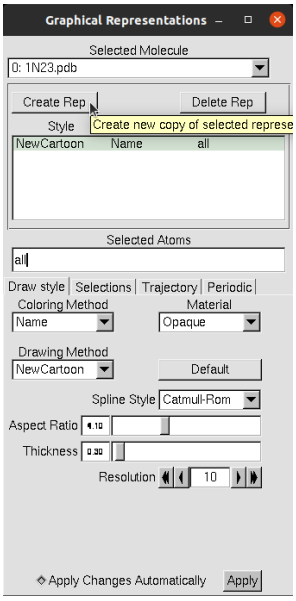

Figure 6: How to create a new representation in VMD.

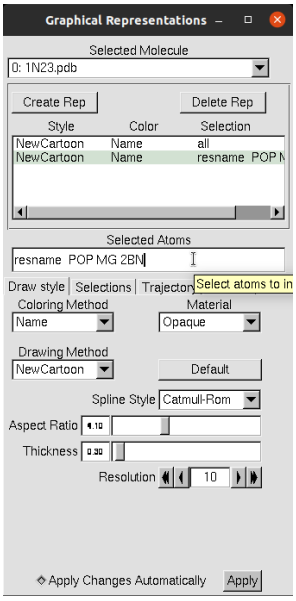

Figure 7: Changing the selected atoms to the 2BN ligand.

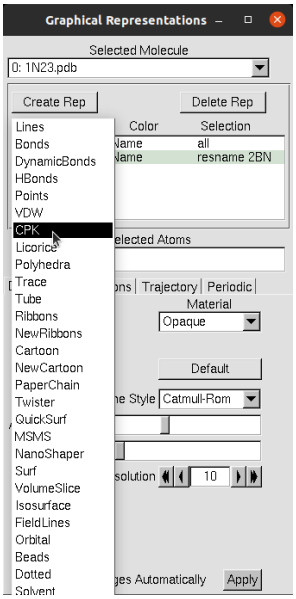

Figure 8: Changing the ligand to the 'CPK' representation.

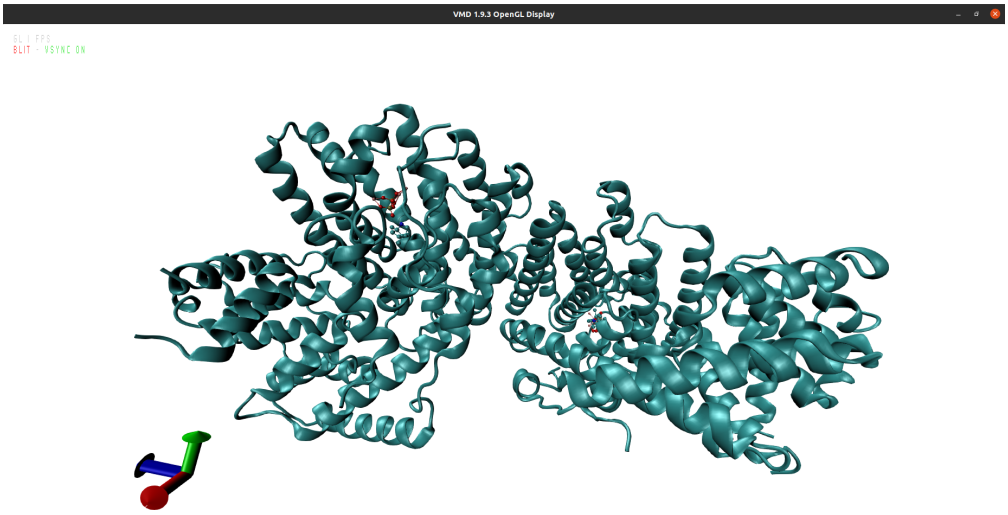

Figure 9: The final render state for visualising the protein and ligand.

Finally, once you have checked that the protein is correct using VMD, you can split the complex pdb '1N23' into its corresponding protein and ligand. You will notice when visualising the structure that it is a dimer complex and so we also want to remove one of the identical protein chains for simplicity. You can do this simply using the `grep` commands in Command 5.3. Finally, check that the files are correct (you will have to manually remove some of the water molecules from the 'B' chain in `vim`) and copy the '2BN.pdb' and '1n23_monomer_NoLig.pdb' files into the Docking directory. This can be done using the '`cp`' command in the Linux terminal.

Command 5.3: How to split the complex crystal structure

bash command

```
$ grep -v " B " > 1n23_monomer.pdb
$ grep "2BN" 1N23_monomer.pdb > 2BN.pdb
$ grep -v "2BN" 1N23_monomer.pdb > 1n23_monomer_NoLig.pdb
```

The coordinates of the ligand that we will be simulating during the workshop can be found, along with any other files required can be found on the GitHub repo for the workshop. The exact path to the Ligand file is https://raw.githubusercontent.com/pcyra2/MD_workshop/main/Files/-Structures/LIG.mol2 and can be downloaded from the terminal using the command `wget`.

Command 5.4: Command to download the ligand structure file.

bash command

```
$ wget ↩
  ↪ https://raw.githubusercontent.com/pcyra2/MD_workshop/main/Files ↩
  ↪ /Structures/LIG.mol2
```

5.b Docking

Task 3: Docking a ligand into the protein.

(a) Dock the ligand

(b) Check docked poses

To run a MD simulation of a ligand-protein complex, you first need a good start structure for the ligand docked into the protein. Luckily many protein structures are obtained with a ligand already docked into the active site and so these positions can be used as a starting point for docking our ligands. We shall be using gnina as it can take the '2BN.pdb' structure generated previously as an initial guess for the active site location.

Command 5.5: Basic input for gnina without GPU acceleration

bash command

```
$ gnina -r 1n23_monomer_NoLig.pdb -l LIG.mol2 --autobox_ligand ↩
  ↪ 2BN.pdb  -o docked.mol2 --no_gpu
```

Option	Value	Type
−r	Receptor/Protein pdb file	Input
−l	Ligand file to be docked	Input
−−autobox_ligand	Stripped receptor/protein parameter file	Input
−o	Output PDB file for docked poses	Output
−−no_gpu	Switches off GPU acceleration (Not necessary if compiled correctly)	Variable

Table 3: The input variables for gnina

The docked poses of the ligand can then be viewed using VMD, like when viewing the original complexed structure. You will however need to load in two files rather than one. It is easiest if you load your '1n23_monomer_NoLig.pdb' file first, then open the 'docked.mol2' file once VMD has loaded (Figures 10 and 11). This will set the docked structures as 'on top' meaning you can cycle through the different docked 'Frames'. You should set your representation for your protein as 'New Cartoon' (Figure 12) and the representation for your docked ligand as 'CPK' (Figure 13).

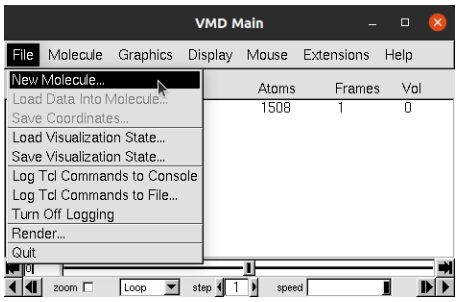

Figure 10: How to load a new molecule in VMD.

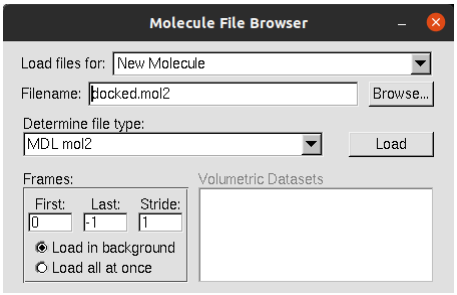

Figure 11: How to select the file.

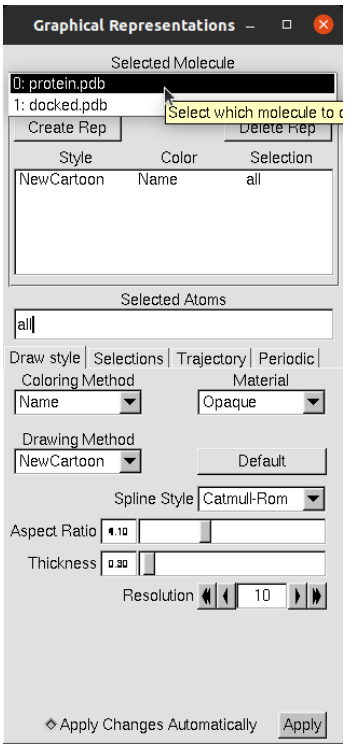

Figure 12: The representation for the protein.

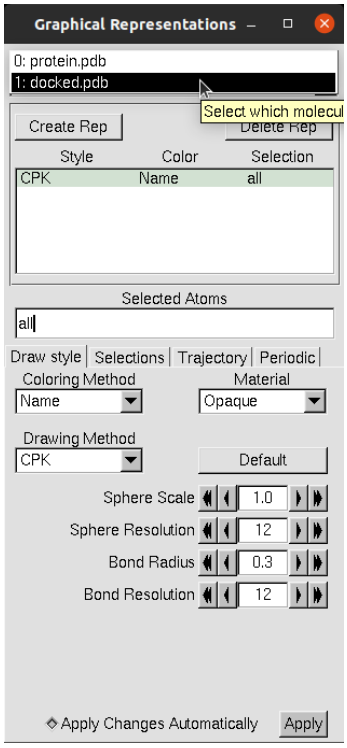

Figure 13: The representation for the docked ligand.

Finally, to use the docked pose for the molecular dynamics simulation you must extract the initial pose from the output coordinate file. You can either do this through the GUI in VMD, or through the CLI. It is recommended to use the CLI as this is the fastest method and ensures that no structural manipulation takes place. To do this you copy the 'docked.mol2' file to 'pose_1.mol2', then open in either vim or nano. From there you can simply delete the other docked poses from the file. (All the lines from the second '@<TRIPOS>MOLECULE'). The final docked file should look something similar to File 5.6.

Output 5.6: Docked pose output file.

bash output

```
@<TRIPOS>MOLECULE
LIG.xyz
 10 9 0 0 0
SMALL
GASTEIGER

@<TRIPOS>ATOM
      1 C  42.7160   46.4533   46.0135 C.3     1  UNL1        0.0090
      2 C  43.7928   47.2054   46.6251 C.3     1  UNL1        0.0224
      3 C  43.7468   48.6707   46.5384 C.3     1  UNL1        0.0047
      4 C  44.7948   45.1707   47.5267 C.2     1  UNL1       -0.0049
      5 C  44.8573   46.4823   47.2270 C.2     1  UNL1       -0.0265
      6 C  41.4018   46.7840   46.8568 C.3     1  UNL1        0.0257
      7 C  41.7288   46.7347   48.3034 C.2     1  UNL1       -0.0346
      8 C  41.7967   45.6383   49.0677 C.2     1  UNL1       -0.0478
      9 C  42.1951   45.7469   50.5025 C.3     1  UNL1        0.0260
     10 C  41.5150   44.2497   48.5990 C.3     1  UNL1        0.0260
@<TRIPOS>BOND
     1     4     5    2
     2     5     2    1
     3     2     3    1
     4     2     1    1
     5     1     6    1
     6     6     7    1
     7     7     8    2
     8     8     9    1
     9     8    10    1
```

You will notice that this docking procedure has removed the protons from our ligand so we shall have to manually add them back in. As our ligand is a carbocation, this process is more

complicated than normal as automating the protonation of the molecule will also neutralise it. One way to fix this is to protonate the ligand to the neutral molecule, then manually remove the proton using a text editor. As 'mol2' files contain bonding information, it is best to do this with a '.xyz' molecule file however.

Command 5.7: Add hydrogen atoms and convert to .xyz.

bash command

```
$ obabel -imol2 pose_1.mol2 -oxyz -Opose_1.xyz -h
$ #This protonates and converts to xyz in one step.
```

Output 5.8: Protonated neutral ligand.

bash output

```
28
LIG.xyz
C         42.71600       46.45330       46.01350
C         43.79280       47.20540       46.62510
C         43.74680       48.67070       46.53840
C         44.79480       45.17070       47.52670
C         44.85730       46.48230       47.22700
C         41.40180       46.78400       46.85680
C         41.72880       46.73470       48.30340
C         41.79670       45.63830       49.06770
C         42.19510       45.74690       50.50250
C         41.51500       44.24970       48.59900
H         42.92737       45.40513       46.05305
H         42.59360       46.71744       44.98387
H         44.18007       47.67066       47.50740
H         43.32922       48.95932       45.59647
H         44.73798       49.06439       46.62478
H         43.13968       49.05599       47.33077
H         45.60368       44.70221       47.96403
H         43.93803       44.63420       47.31901
H         45.72789       46.99089       47.44709
H         40.64322       46.06240       46.63601
H         41.04477       47.75962       46.60067
H         41.92334       47.63624       48.76643
H         42.37555       46.77311       50.74586
H         43.08636       45.17877       50.66917
H         41.40891       45.36665       51.12073
H         41.23917       44.27121       47.56539
H         40.71320       43.83510       49.17356
H         42.39066       43.64721       48.72201
```

When visualising the file, you will see that there is a proton on the carbocation. Selecting this atom shows us that it is atom index 12 and as `vmd` is 'zero indexed', it is the 13th atom which needs removing. Delete this line in the 'pose_1.xyz' file, and change the number at the top from 28 to 27 as this is the number of atoms in the molecule. Finally, we will convert back to 'mol2' using obabel and Command 5.9.

Command 5.9: Convert back to mol2.

bash command

```
$ obabel -ixyz pose_1.xyz -omol2 -Opose_1.mol2
```

Finally, you need to check that the hybridisation states for all of the carbons are correct. There should be 5 'C.2' carbons as there are 2 double bonds and 1 cationic carbon atom. It is likely that obabelen incorrectly identified the carbocation as 'C.2' and so you should manually change this in your file (the carbocation should be carbon 2). The final file should look something like Output 5.10 however we have excluded the bonding information for simplicity.

Output 5.10: The final structure for the docked molecule.

bash output

```
@<TRIPOS>MOLECULE
LIG.xyz
 27 26 0 0 0
SMALL
GASTEIGER

@<TRIPOS>ATOM
     1 C  42.7160   46.4533   46.0135 C.3     1  UNL1    -0.0400
     2 C  43.7928   47.2054   46.6251 C.2     1  UNL1    -0.0052
     3 C  43.7468   48.6707   46.5384 C.3     1  UNL1    -0.0553
     4 C  44.7948   45.1707   47.5267 C.2     1  UNL1    -0.1021
     5 C  44.8573   46.4823   47.2270 C.2     1  UNL1    -0.0844
     6 C  41.4018   46.7840   46.8568 C.3     1  UNL1    -0.0337
     7 C  41.7288   46.7347   48.3034 C.2     1  UNL1    -0.0853
     8 C  41.7967   45.6383   49.0677 C.2     1  UNL1    -0.0796
     9 C  42.1951   45.7469   50.5025 C.3     1  UNL1    -0.0440
    10 C  41.5150   44.2497   48.5990 C.3     1  UNL1    -0.0440
    11 H  42.9274   45.4051   46.0530 H       1  UNL1     0.0277
    12 H  42.5936   46.7174   44.9839 H       1  UNL1     0.0277
    13 H  43.3292   48.9593   45.5965 H       1  UNL1     0.0238
    14 H  44.7380   49.0644   46.6248 H       1  UNL1     0.0238
    15 H  43.1397   49.0560   47.3308 H       1  UNL1     0.0238
    16 H  45.6037   44.7022   47.9640 H       1  UNL1     0.0532
    17 H  43.9380   44.6342   47.3190 H       1  UNL1     0.0532
    18 H  45.7279   46.9909   47.4471 H       1  UNL1     0.0572
    19 H  40.6432   46.0624   46.6360 H       1  UNL1     0.0308
    20 H  41.0448   47.7596   46.6007 H       1  UNL1     0.0308
    21 H  41.9233   47.6362   48.7664 H       1  UNL1     0.0571
    22 H  42.3755   46.7731   50.7459 H       1  UNL1     0.0274
    23 H  43.0864   45.1788   50.6692 H       1  UNL1     0.0274
    24 H  41.4089   45.3666   51.1207 H       1  UNL1     0.0274
    25 H  41.2392   44.2712   47.5654 H       1  UNL1     0.0274
    26 H  40.7132   43.8351   49.1736 H       1  UNL1     0.0274
    27 H  42.3907   43.6472   48.7220 H       1  UNL1     0.0274
@<TRIPOS>BOND
 # Bonding information
```

5.c Molecular Dynamics Preparation

Task 4: Preparing for a dynamics simulation.

(a) Parameterise the ligand

(b) Generate topology and parameter files

5.c.i Parameterisation

To run a molecular dynamics simulation, you first need to parameterise all of the atoms in the system. There are multiple different molecular dynamics force fields that contain standard parameters for common amino acids and some other small or simple molecules. For bigger or more complicated molecules, however, you will need to create your own custom parameters.

Although there is a rigorous method that can be undertaken to correctly parameterise complex ligands, including comparing the parameters with *ab-initio* calculations, we shall be using an automated method for speed and simplicity. [a]

We shall be using the `acpype`(*8*) automated script to parameterise the ligands in this workshop, which combines the parameterisation tools offered by Amber into a simple-to-use script. We shall be using `AM1-BCC`(*16*) charges and the `gaff`(*17*) atom typing so that it will be compatible with the 'FF14SB' forcefield(*3*) that we shall be using for the simulation.

Command 5.11: How to use acpype.

bash command

```
$ conda activate acpype
$ acpype -i pose_1.mol2 -c bcc -n 1 -a gaff
```

[a] You can learn more about the rigorous process of parameterisation here.

This will generate a `pose_1.acpype` folder containing parameter and coordinate files in a variety of different file types including those required for CHARMM(*9*) and GROMACS(*10*). The files that we want for this workshop however are the forcefield modification file which allows us to add parameters to the existing FF14SB(*3*) forcefield, as well as the `mol2` file containing the newly calculated `AM1-BCC` atomic charges. The two files should be called `pose_1_AC.frcmod` and `pose_1_bcc_gaff.mol2` and should be copied into the directory above.

5.c.ii Topology and coordinate setup

The next step in initiating any molecular dynamics simulation is to generate your coordinate and topology files that will be used in the calculation. We split topology and coordinates into two files rather than one due to the expectation that the coordinates for the atoms will change as a simulation progresses, however, the parameters for each of the atoms will not. This allows for future coordinate and trajectory files to be much smaller in size due to them only containing positional information for the system.

Currently, you have a `pdb` structure file for your protein-ligand complex and `mol2` structure file for your ligand containing charge parameters. We need to combine these files together and solvate them in order to simulate the system in biological conditions. The program that we use to do this is `tleap` which is part of the AmberTools suite of software.

Before running tleap however, it is important to know that it can sometimes struggle to load in a protein file that also has non-standard molecules in so it is good practice to break down the '1n23_monomer_NoLig.pdb' further. This can be done using a combination of `grep` to extract the coordinates of the 'POP' and 'MG' residues, then use `pdb4amber` to strip the residues from the structure file.

Command 5.12:

bash command

```
$ pdb4amber -i 1n23_monomer_NoLig.pdb -o protein.pdb -s :MG,POP
```

The acpype tool is useful for changing the atom types of other small molecules within our protein. Although the parameters for diphosphate are already known to Amber, we can ensure they are used by setting the atom type of the diphosphate molecule to 'amber' and generating a new 'mol2' file using these atom types. Make sure to also specify the correct charge for this molecule and then copy the POP_bcc_amber.mol2 file into the working directory also.

Once you have split the '1n23_monomer_NoLig.pdb' file into the 'protein.pdb', 'POP_bcc_amber.mol2' and 'MG.pdb' files, you can then run tleap using the input File 5.1.

File 5.1: Input file required for the tleap program.

tleap file

```
source leaprc.protein.ff14SB        # Loads general protein parameters
source leaprc.water.tip3p           # Loads water parameters
source leaprc.gaff                  # Loads "gaff" parameters

LIG = loadmol2 LIG_bcc_gaff.mol2
# Loads the coordinates and charges for the ligand

loadamberparams LIG_AC.frcmod
# Loads the new parameters for the ligand

check LIG
# Checks that all parameters are present for the ligand

MG = loadpdb "MG.pdb"
```

```
check MG
# Loads the active site counterions

POP = loadmol2 POP_bcc_amber.mol2
check POP
# Loads the active site diphosphate

PROT = loadpdb "PROTEIN.pdb"
# Loads the protein structure

check PROT
# Checks for missing parameters for your protein

COMP = combine {PROT MG POP LIG}
# Combines the protein and ligand

solvateoct COMP TIP3PBOX 10.0
# Adds an octahedral box of water automatically with a minimum
↪  distance of 10 A from the protein

addions COMP Na+ 0
# Neutralises the net charge of the system by adding counterions

saveamberparm COMP complex.parm7 start.rst7
#Saves parameters and initial coordinates

quit
```

5.d Molecular Dynamics

Task 5: Run a molecular dynamics simulation.

(a) Generate input files for:

(i) Minimisation

(ii) Heating

(iii) Equilibration

(iv) Production dynamics

(b) Run the simulations for:

(i) Minimisation

(ii) Heating

(iii) Equilibration

(iv) Production dynamics

Once we have correctly generated our initial start coordinate system and the corresponding parameters, we can begin to run some dynamics simulations. A traditional dynamics simulation starts with a minimisation step which allows for any steric clashes or small structural problems within our initial system to be corrected before trying to introduce time. These steric clashes are very common as when you solvate the system, the water molecules are randomly placed and so can sometimes lead to problems if not fixed. It is common to run a multi-stage minimisation where you initially fix all atoms except for solvents and minimise, before slowly relaxing these restraints until a full system minimisation is performed. In the interest of time, however, we shall not be doing this.

File 5.2: Minimisation input file.

sander file

```
initial minimisation script at 0k
&cntrl
  imin=1,       # initiate minimisation
  ntx=1,        # read coordinates with no velocities
  irest=0,      # don't restart the simulation
  maxcyc=10000, # 10000 minimisation steps
  ncyc=5000,    # 5000 steps of steepest gradient
  ntpr=10,      # print to mdout every 10 steps
  ntwx=0,       # don't print trajectory file
  cut=8.0,      # non-bonded cut off of 8 A
/
```

Firstly we need to create an input file that tells amber what type of calculation we want to perform. These input files have many variables, however, in a lot of cases the default values are fine. We will first generate a simple minimisation input file called 'min.in' that should look like File 5.2 and should be saved to the `Dynamics` subdirectory. The 'start.rst7' and 'complex.parm7' files generated from the previous step should also be copied here. Finally, to run the initial minimisation, use Command 5.13.

Command 5.13: Basic amber command to run the minimisation input file.

bash command

```
$ # If using the conda install or have no mpi compiled version:
$ sander -O -i min.in -o min.out -c start.rst7 -p complex.parm7 ↩
  ↪ -inf min.mdinfo -r min.ncrst
$
$ # If compiled with mpi:
$ mpirun -np NUM_CORES sander.MPI -O -i min.in -o min.out -c ↩
  ↪ start.rst7 -p complex.parm7 -inf min.mdinfo -r min.ncrst
$
$ # If using the full version of amber with GPU support:
$ pmemd.cuda -O -i min.in -o min.out -c start.rst7 -p complex.parm7 ↩
  ↪ -inf min.mdinfo -r min.ncrst
$ ### NOTE: Often the pmemd version struggles if large clashes are ↩
  ↪ present. If you get an error, try switching to sander.
```

It should be noted that there are two main programs for running molecular dynamics within amber: 'sander' and 'pmemd'. If you are using the free version of amber (AmberTools), then you will only have access to 'sander', however, if you have the full version then you will also have 'pmemd'. These two programs are functionally similar, with `pmemd` being an optimised and slimmed down code which can be run on GPUs where as `sander` is less optimised but much more rigorous. Both programs can be `mpi` accelerated, however, so even with the free version of amber you can run some simple dynamics simulations. Some functionality of `sander` is lost when changing to `pmemd` however so always check to see if your calculation will work with your code.

Option	Value	Type
–O	Overwrite previous files	Variable
–i	Input file	Input
–o	Main output file	Output
–c	Simulation start coordinates	Input
–p	Parameters for your system	Input
–inf	Dynamics information about current step	Output
–r	'Restart' file for the latest coordinates to be saved	Output
–x	'Trajectory' file for saving the coordinates every 'ntwx' steps	Output

Table 4: The input variables for sander or pmemd

Once minimisation is complete, the next step is to heat the system. Currently, the temperature is at 0 K, however, we need this to be at physiological temperature. This exact temperature can vary depending on what exactly you are simulating however for this workshop we shall heat the system to 300 K. The temperature of the system is determined by the average kinetic energy for all atoms and is related to temperature using the Maxwell-Boltzman distribution. The temperature of the system is controlled by a thermostat which modifies the velocities of the atoms in order to keep the system at the desired temperature.

To run the heating simulation, you first need to generate the input file and this should look similar to File 5.3. You shall notice that we are now using more variables in the `&cntrl` section and also have introduced an `&wt` section which allows us to vary settings throughout a simulation. Once you have generated your input file, you can run the simulation in a similar way to before, however, use the restart file from the minimisation step as your start coordinates. You also need to define the trajectory output file for this simulation as you will be printing the trajectory (Command 5.14).

File 5.3: Heat input file.

sander file

```
heating script from 0 k to 300 k over 200 ps
&cntrl
  imin=0,           # run a dynamics simulation
  ntx=1,            # read coordinates with no velocities
  irest=0,          # don't restart the simulation
  nstlim=100000,    # run simulation for 100000 steps
  dt=0.002,         # each step is separated by 0.002 ps (200 ps total)
  ntf=2, ntc=2,     # constrain bonds with hydrogen
  ntpr=100,         # print to mdout every 100 steps
  ntwx=100,         # print trajectory file every 100 steps
  cut=8.0,          # non-bonded cut off of 8 A
  ntb=1,            # constant volume with periodic boundary conditions
  ntp=0,            # no pressure control
  ntt=3,            # control temperature using Langevin Dynamics
  gamma_ln=2.0,     # Langevin collision frequency
  nmropt=1,         # control NMR restraints
  ig=-1,            # use a random seed
/
&wt type='TEMP0', istep1=0, istep2=100000, value1=0.0, value2=300.0 /
&wt type='END' /
# Heat the system from 0 to 300 k from step 0 to end
```

Command 5.14: Basic amber command to run the heating input file.

bash command

```
$ sander -O -i heat.in -o heat.out -c min.rst7 -p complex.parm7 ↩
  ↪ -inf heat.mdinfo -r heat.ncrst -x heat.nc
$
$ # You can use sander.MPI or pmemd.cuda instead of sander.
```

The final two steps are to equilibrate the system and then run the simulation using the Constant number of atoms, volume and pressure (NVP) ensemble. It is important to note that equilibration and production dynamics are functionally identical. The equilibration stage is simply to ensure that any of the restraints on the system that are imposed are equilibrated and representative of the real system before collecting data on them. In theory, you do not need to split these into separate calculations, however, it is useful for analysis as you then simply exclude equilibration data from your post-processing techniques.

File 5.4: Equil input file.

sander file

```
equilibration script at 300k for 200 ps
&cntrl
  imin=0,           # run a dynamics simulation
  ntx=5,            # read coordinates with no velocities
  irest=1,          # don't restart the simulation
  nstlim=100000,    # run simulation for 100000 steps
  dt=0.002,         # each step is separated by 0.002 ps (200 ps total)
  ntf=2, ntc=2,     # constrain bonds with hydrogen
  ntpr=100,         # print to mdout every 100 steps
  ntwx=100,         # print trajectory file every 100 steps
  cut=8.0,          # non-bonded cut off of 8 A
  ntb=2,            # constant volume with periodic boundary conditions
  ntp=1,            # pressure control
  ntt=3,            # control temperature using Langevin Dynamics
  barostat=1,       # Berendsen barostat for pressure control
  gamma_ln=2.0,     # Langevin collision frequency
  ig=-1,            # use a random seed
  temp0=300.0       # start at 300 k
  tempi=300.0       # end at 300 k (Constant temp)
/
```

To run the dynamics simulation, simply copy your 'equil.in' file to 'prod.in' and change the number of steps from 100000 to 500000 steps. This will create 1 ns of production dynamics simulation in total. You can run both of these simulations in the same way as the heating simulation, remembering to change the start coordinate file to the final restart file of the previous simulation.

5.e Analysis

Task 6: Analysing the simulations

(a) Plot:

(i) Energy of minimisation

(ii) Heating temperature

(b) Convert trajectories if on windows (optional)

(c) Visualise production dynamics trajectory

5.e.i Basic Analysis

To extract basic information from our dynamics output files, there are two automated tools that are provided by amber. '`process_minout.perl`' is used exclusively for analysis of minimisation outputs and '`process_mdout.perl`' is used for all other dynamics simulation outputs. They are both simple perl scripts that extract all of the data from within the output files and format them as plottable data files.

To use these scripts, firstly create an 'analysis' directory within your 'dynamics' directory. Then create a 'min' directory within the 'analysis' directory and move into it. From there, run the Command 5.15.

Command 5.15: Running process_mdout.perl.

bash command

```
$ process_minout.perl ../../min.out
```

Command 5.16: Plotting a 2D graph using xmgrace (Linux).

bash command

```
$ xmgrace summary.ENERGY
```

This should create a directory filled with many output files, but the file we are most interested in is the 'summary.ENERGY' file. This is a data file containing the system's total energy with respect to the minimisation step. We can plot this (Figure 14) using `xmgrace` (Command 5.16) to understand whether our simulation minimised sufficiently before starting dynamics simulations.

If your energy seems to plateau towards the end then you can assume that the simulation has sufficiently minimised and carry on with further dynamics simulations. You should note that the energy plotted here has no real-world uses and is heavily dominated by the solvent molecules in your system. Raw energy data should not be used for calculating the binding of molecules.

Next, create a 'simulation' directory within your analysis section. Then run the '`process_mdout.perl`' command in a similar way to the 'minout' script in the directory, ensuring to load the 'heat.out', 'equil.out' and 'prod.out' files. We are initially interested in the 'summary.TEMP' file, which contains the temperature of the system against time in picoseconds (PS). This should linearly increase from 0 K to 300 K for the first 200 ps and then remain constant for the rest of the simulation (Figure 15). This is because the equilibration and production simulations were run under the Constant number of atoms, volume and temperature (NVT) ensemble with constant volume and temperature.

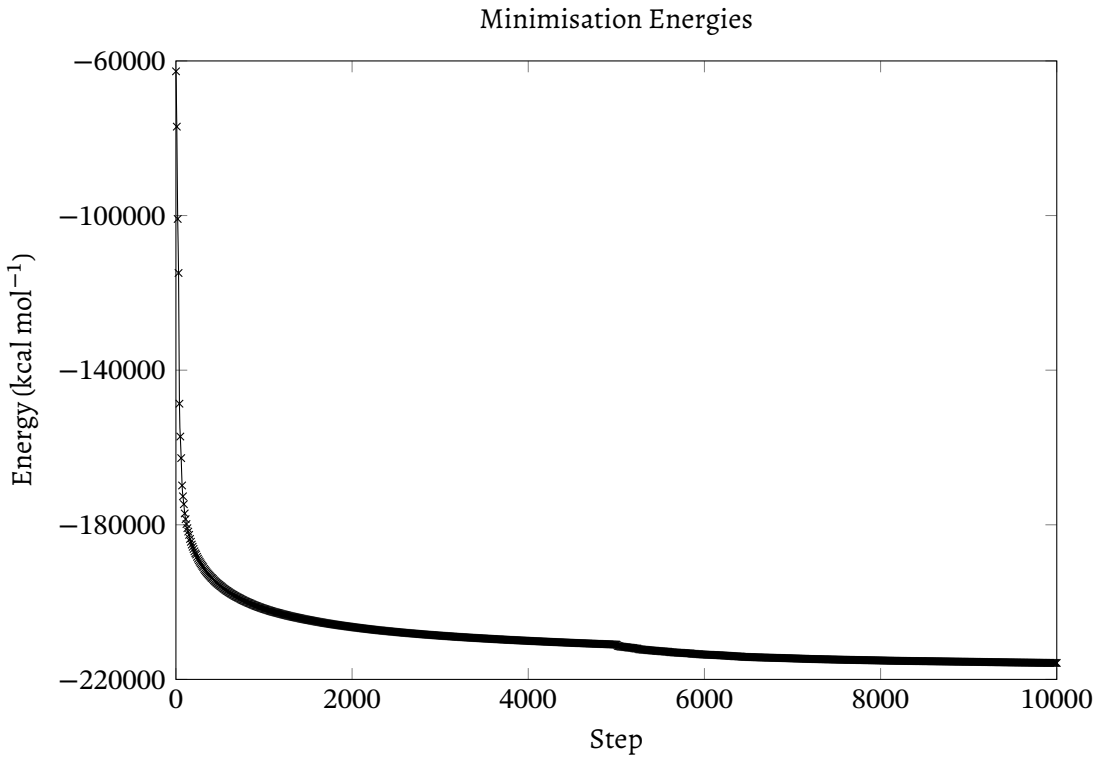

Figure 14: A plot of the system's total energy with respect to minimisation steps.

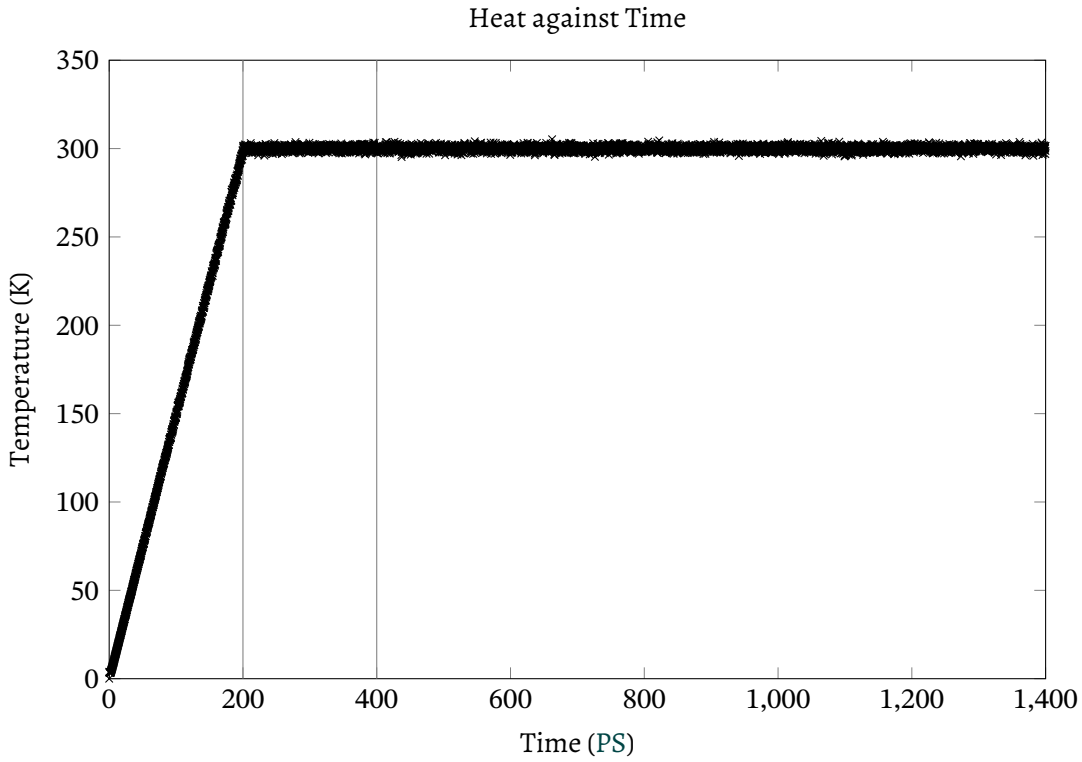

Figure 15: A plot of the heat of the system with respect to time (PS).

Next, we are interested in seeing if the volume is constant throughout the equilibration and production dynamics simulations. To check this, you first need to edit the 'summary.VOLUME' file and remove the first 200 ps of data. This is due to the heating simulation not containing volume information and so `xmgrace` cannot understand this part of the data file. Once the heating data is removed, the plot will start at a large volume and should quickly equilibrate to a constant volume. This volume should then stay constant for the rest of the simulation like in Figure 16. The sudden drop in volume is one of the reasons why an equilibration stage is necessary as when you initially start volume controls, the volume needs some time to equilibrate.

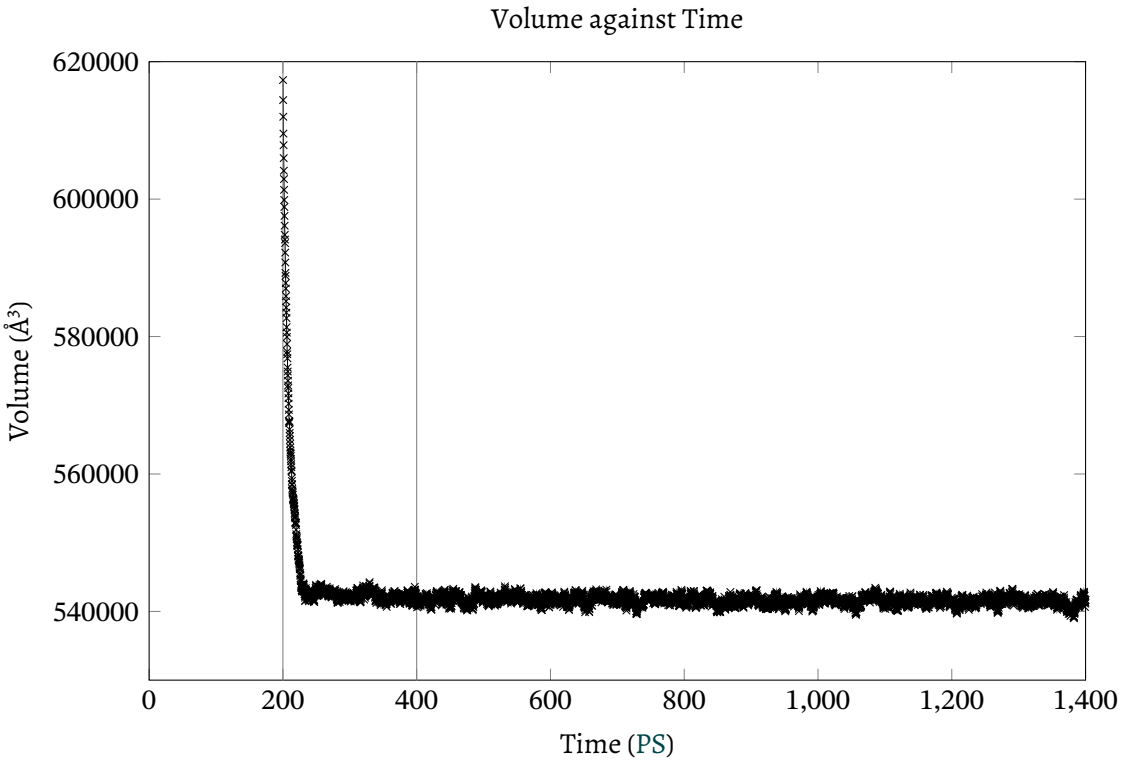

Figure 16: A plot of the volume of the system with respect to time (PS).

Finally, you should check that the energies of the system stabilise before the production stage of the simulation. As the energy of the system is split into its constituents, you can plot the total energy, potential energy and kinetic energy and check that all three converge correctly (Command 5.17).

Command 5.17: Plotting the energies of the system

bash command

```
$ xmgrace summary.ETOT summary.EPTOT summary.EKTOT
```

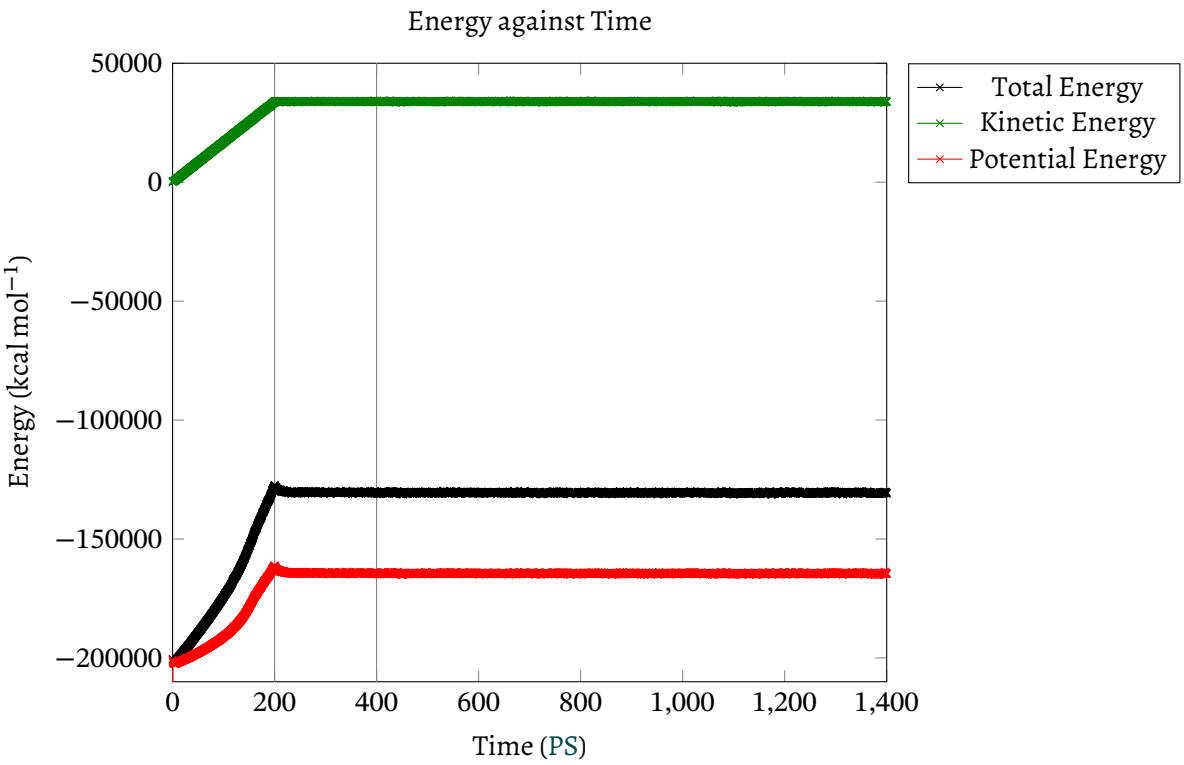

Figure 17: A plot of the energy of the system with respect to time (PS).

5.e.ii Cpptraj

Once you have used the automated scripts to do the basic analysis of the simulations, further insight can be gained by performing analysis on your trajectories. '`cpptraj`' can be used as a trajectory analysis tool and this can be run interactively in the terminal, or through use of an input file.

One thing that is often interesting to measure is the `root mean square deviation (RMSD)` of the atoms throughout the simulation. This is a way of measuring the structural change of the atoms measured with respect to a reference structure. RMSD analysis is often performed on the backbone atoms only, and can be performed in cpptraj using File 5.5. Cpptraj can then be run using Command 5.18 and the resulting data can be plotted and should look similar to Figure 18.

File 5.5: RMSD backbone analysis input file.

cpptraj file

```
parm ../../complex.parm7
# Loads the parameter file

trajin ../../heat.nc
trajin ../../equil.nc
trajin ../../prod.nc
# Loads the trajectories


rmsd backbone_RMSD @C,CA,N out backbone_RMSD.dat
run
```

Command 5.18: Running cpptraj with a file.

bash command

```
$ cpptraj cpptraj.in
```

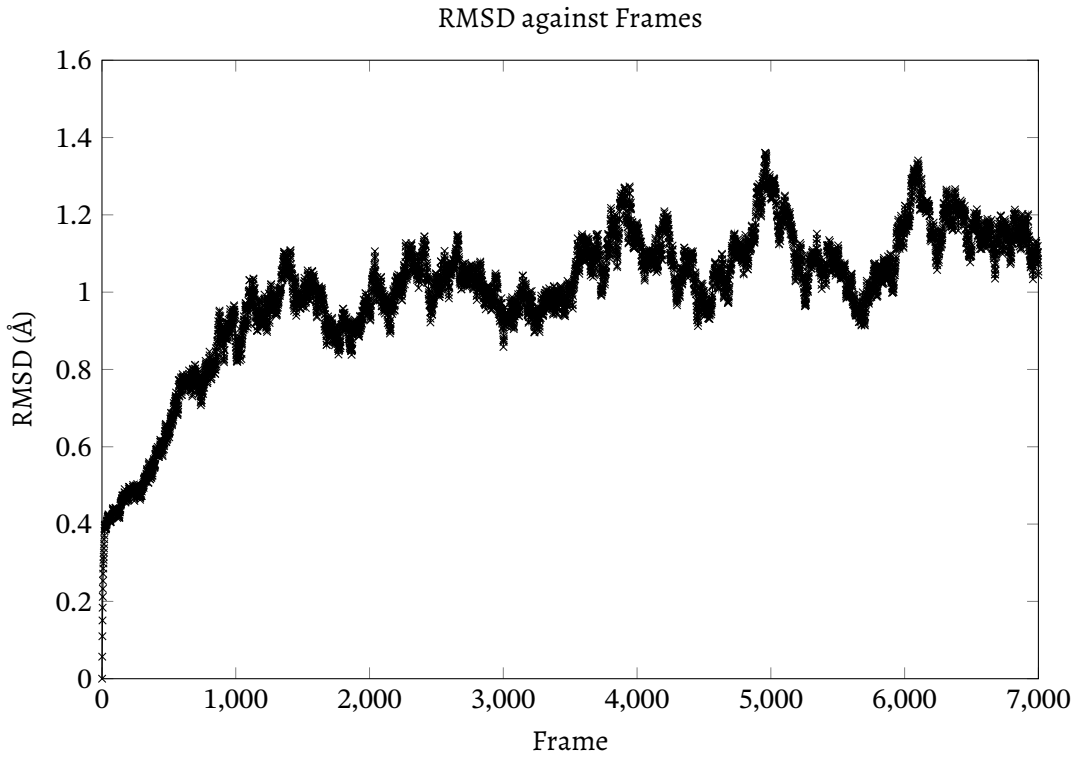

Figure 18: A plot of the RMSD of the system with respect to trajectory frames.

It should now be noted that the x-axis is no longer in units of time and instead trajectory frames. As our 'ntwx' parameter was set to 100 however for the heating, equilibration and production dynamics you can convert to time by using Equation (5.1). Although the RMSD of the backbone doesn't fully plateau, some fluctuations are expected as the protein is naturally going to vibrate within nature at 300 K. A RMSD of less than 1.5 is usually considered good for a simulation.

$$\begin{aligned} Time\,(ps) &= Frames \times \Delta T \times ntwx \\ \Delta T &= 0.002\,(ps) \\ ntwx &= 100 \end{aligned} \tag{5.1}$$

5.e.iii Visualising the trajectory

One other piece of analysis that you should always perform is to visually inspect the trajectory. It is impossible to calculate the binding affinity of a molecule if it leaves the active site during the simulation and so visually checking that nothing strange is happening is an essential part of checking your data. If working on Linux, this is incredibly easy and can be done using VMD. You simply need to load the trajectory and parameter files in, like how you would visualise a restart coordinate file Command 5.19.

Command 5.19: Visualising the trajectory in linux.

bash command

```
$ vmd prod.nc complex.parm7
```

If on Windows, however, you cannot load '.nc' files into vmd. You instead need to convert them to a '.binpos' file. To do this, you can again use cpptraj and File 5.6 to obtain the new trajectory file. This can be then loaded into vmd like any other coordinate file. This should be run in the main 'Dynamics' directory.

File 5.6: Convert .nc to .binpos.

cpptraj file

```
parm complex.parm7
trajin prod.nc

trajout prod.binpos
```

5.f Free Energy calculation

Task 7: Calculating the free energy of binding.

(a) Extract the receptor, ligand and complex parameter files

(b) Perform a MM/GBSA calculation

There are many different ways of calculating the Gibbs free energy of binding for a ligand into a protein(18–22), however, the simplest and quickest way is to use either the MM/GBSA or MM/PBSA method.(22) The scientific theory behind these methods can be found in Section 2.c.

There are two different ways of performing these types of calculations. The first method is to run independent simulations of the ligand, protein and complex in order to calculate the $\Delta G_{\text{Solv, Lig.}}$, $\Delta G_{\text{Solv, Rec.}}$ and $\Delta G_{\text{Solv, Complex}}$ directly, and then calculate $\Delta G_{\text{Vaccum, Bind.}}$ from the interactions between the ligand and protein in the complex simulation. The other way of performing this calculation is to perform a single simulation of the complexed system. It is then possible to separate each of the energies out post-hoc. The advantages of this method are that it only requires a single simulation, meaning it is much less computationally intensive so you can potentially run a longer simulation. This type of calculation also has been shown to increase accuracy. Due to the ease and accuracy of the single simulation method, we are going to use this for calculating the binding energy of the ligand.

5.f.i Setup

Firstly, you need to copy the production-dynamics trajectory file from the previous task to the 'Free_Energy's directory, along with the parameter file for the solvated complex. You should then rename the parameter file to indicate that it is solvated, as you are about to remove the solvent in the next step. You can then use the `ante-MMPBSA.py` script to prepare the stripped parameter files for the complex, receptor and ligand respectively.

Command 5.20: Commands to setup the GBSA calculation.

bash command

```
$ ante-MMPBSA.py -p SOLVATED_COMPLEX.parm7 -c complex.parm7 -r
  ↪ receptor.parm7 -l ligand.parm7 -s :WAT,Na+ -n :LIGAND_NAME
  ↪ --radii=mbondi2
```

The input variables for the `ante-MMPBSA.py` are as follows:

Option	Value	Type
-p	Solvated complex parameter file	Input
-c	Stripped complex parameter file	Output
-r	Stripped receptor/protein parameter file	Output
-l	Stripped ligand parameter file	Output
-s	Strip mask for the solvent and counter ions.	Variable
-radii	Atomic Radii parameter set	Variable

5.f.ii Calculation

Once the stripped parameter files are generated, one final input file is required in order to run the MM/GBSA or MM/PBSA calculation. This file has a similar format to the input files used by `sander` and other AMBER programs.

File 5.7: Input file for required for the MM/GBSA program.

`MMPBSA.py` file

```
GBSA input file
&general
```

```
    startframe=1, endframe=5000, interval=20, \\ This allows you to
    ↪  speed up the calculation by only calculating every n bins.
    strip_mask=:WAT:Na+, \\ The AMBER mask for the solvent and
    ↪  counter ions that were stripped in the previous step.
    keep_files=0, \\ Delete temporary files after the calculation
    ↪  completes.
/
&gb
    igb=2, \\ Specifies the generalised Born method to use.
    saltcon=0.100, \\ Salt concentration for the implicit solvent (M)
/
```

Once all the files are in place, the MM/GBSA calculation can begin. Use the following command with your file names to run the calculation. Make sure you change the variables:

- `NUM_PROCESSORS` to the number of available threads on your machine
- `INPUT_FILE` to the name of your newly created MM/GBSA input file
- `SOLVATED_COMPLEX` to the name of your original solvated complex parameter file
- `PRODUCTION_TRAJECTORY` to the name of the production trajectory file

Command 5.21: Commands to run the GBSA calculation.

bash command

```
$ mpirun -np NUM_PROCESSORS MMPBSA.py.MPI -O -i INPUT_FILE.in -o
    ↪  gb_results.dat -sp SOLVATED_COMPLEX.parm7 -cp complex.parm7
    ↪  -rp receptor.parm7 -lp ligand.parm7 -y
    ↪  PRODUCTION_TRAJECTORY.nc
```

5.f.iii Analysis

The data from the MM/GBSA calculation is outputted to the 'gb_results.dat' file. Within this file, the free energies of $\Delta\bar{G}_{complex}$, $\Delta\bar{G}_{prot.}$ and $\Delta\bar{G}_{lig.}$ required for Equation (2.4) are printed and the final value of ΔG_{bind} is shown on the last line. The standard deviation and standard error of each value are also printed.

Output 5.22: End of the gb_results.dat file

bash output

```
Ligand:
Energy Component       Average        Std. Dev.   Std. Err. of Mean
------------------------------------------------------------------
VDWAALS                -0.1898           0.9840              0.0622
EEL                    49.6063           0.6127              0.0388
EGB                   -52.5867           0.2998              0.0190
ESURF                   2.3168           0.0365              0.0023

G gas                  49.4165           1.1792              0.0746
G solv                -50.2699           0.3088              0.0195

TOTAL                  -0.8534           1.1370              0.0719

Differences (Complex - Receptor - Ligand):
Energy Component       Average        Std. Dev.   Std. Err. of Mean
------------------------------------------------------------------
VDWAALS               -27.5565           1.7393              0.1100
EEL                  -298.4838           8.5471              0.5406
EGB                   294.7172           8.2767              0.5235
ESURF                  -3.9973           0.0728              0.0046

DELTA G gas          -326.0403           8.3852              0.5303
DELTA G solv          290.7198           8.2648              0.5227

DELTA TOTAL           -35.3204           2.2383              0.1416


------------------------------------------------------------------
```

6 Cheat codes

If you have any problems with any of the tasks within this workshop, here is a list of commands which should help you to complete the task. We recommend using this as a last resort, however, as simply copying and pasting commands doesn't allow you to learn the methodology.

6.a Task 1: Initiating the file system

Command 6.1: Linux commands to initiate your local file structure.

bash command

```
$ mkdir MD_Workshop
$ cd MD_Workshop
$ mkdir Structures
$ mkdir Docking
$ mkdir Dynamics
$ mkdir Free_Energy
```

6.b Task 2: Getting the structures

Command 6.2: Linux commands to obtain the structures.

bash command

```
$ cd Structures
$ wget https://files.rcsb.org/download/1N23.pdb
$ wget ↩
  ↪ https://raw.githubusercontent.com/pcyra2/MD_workshop/main/Files ↩
  ↪ /Structures/LIG.mol2
$ grep -v " B " 1N23.pdb > 1n23_monomer.pdb
$ grep "2BN" 1n23_monomer.pdb > 2BN.pdb
$ grep -v "2BN" 1n23_monomer.pdb > 1n23_monomer_NoLig.pdb
$ grep -v "HOH B" 1n23_monomer_NoLig.pdb > tmp
$ mv tmp 1n23_monomer_NoLig.pdb
```

6.c Task 3: Docking the ligand

Command 6.3: How to dock the ligand into the protein.

bash command

```
$ cd ../Docking
$ cp ../Structures/1n23_monomer_NoLig.pdb .
$ cp ../Structures/2BN.pdb .
$ cp ../Structures/LIG.mol2 .
$ gnina -r 1n23_monomer_NoLig.pdb -l LIG.mol2 --autobox_ligand ↩
  ↪ 2BN.pdb -o docked.mol2 --no_gpu
$ head -n 27 docked.mol2 > pose_1.mol2
$ obabel -imol2 pose_1.mol2 -oxyz -Opose_1.xyz -h
$ sed -i.bak -e '15d' pose_1.xyz
$ sed -i.bak "1s/28/27/g" pose_1.xyz
$ obabel -ixyz pose_1.xyz -omol2 -Opose_1.mol2
$ sed -i.bak '9s/C.3/C.2/g' pose_1.mol2
```

6.d Task 4: Setting up a dynamics simulation

Command 6.4: How to set up your simulations.

bash command

```
$ cd ../Dynamics
$ mkdir setup
$ cd setup
$ cp ../../Docking/pose_1.mol2 .
$ cp ../../Docking/1n23_monomer_NoLig.pdb .
$ conda activate acpype
$ acpype -i pose_1.mol2 -c bcc -n 1 -a gaff
$ cp pose_1.acpype/pose_1_AC.frcmod .
$ cp pose_1.acpype/pose_1_bcc_gaff.mol2 .
$ grep "POP" 1n23_monomer_NoLig.pdb > POP.pdb
$ grep "MG" 1n23_monomer_NoLig.pdb > MG.pdb
$ acpype -i POP.pdb -c bcc -n -4 -a amber
$ cp POP.acpype/POP_bcc_amber.mol2 .
$ wget ↩
  ↪ https://raw.githubusercontent.com/pcyra2/MD_workshop/main/Files ↩
  ↪ /MDPrep/leap.in
$ pdb4amber -i 1n23_monomer_NoLig.pdb -o protein.pdb -s :MG,POP
$ conda activate AmberTools
$ tleap -f leap.in
```

6.e Task 5: Running a dynamics simulations

Command 6.5: Running the simulations.

bash command

```
$ cp start.rst7 ../
$ cp complex.parm7 ../
$ cd ../
$ wget ↩
  ↪ https://raw.githubusercontent.com/pcyra2/MD_workshop/main/Files ↩
  ↪ /Dynamics/min.in
$ wget ↩
  ↪ https://raw.githubusercontent.com/pcyra2/MD_workshop/main/Files ↩
  ↪ /Dynamics/heat.in
$ wget ↩
  ↪ https://raw.githubusercontent.com/pcyra2/MD_workshop/main/Files ↩
  ↪ /Dynamics/equil.in
$ wget ↩
  ↪ https://raw.githubusercontent.com/pcyra2/MD_workshop/main/Files ↩
  ↪ /Dynamics/prod.in
$ sander -O -i min.in -o min.out -c start.rst7 -p complex.parm7 -r ↩
  ↪ min.ncrst -inf min.mdinfo
$ sander -O -i heat.in -o heat.out -c min.ncrst -p complex.parm7 -r ↩
  ↪ heat.ncrst -inf heat.mdinfo -x heat.nc
$ sander -O -i equil.in -o equil.out -c heat.ncrst -p complex.parm7 ↩
  ↪ -r equil.ncrst -inf equil.mdinfo -x equil.nc
$ sander -O -i prod.in -o prod.out -c equil.ncrst -p complex.parm7 ↩
  ↪ -r prod.ncrst -inf prod.mdinfo -x prod.nc
```

6.f Task 6: Analysis of the simulations

Command 6.6: Runnning analysis.

bash command

```
$ mkdir analysis
$ cd analysis
$ mkdir min
$ cd min
$ process_minout.perl ../../min.out
$ cd ../
$ mkdir simulation
$ cd simulation
$ process_minout.perl ../../heat.out ../../equil.out ../../prod.out
$ tail -n 6000 summary.VOLUME > tmp
$ mv tmp summary.VOLUME
$ wget ↩
  ↪ https://raw.githubusercontent.com/pcyra2/MD_workshop/main/Files ↩
  ↪ /Analysis/cpptraj.in
$ cpptraj cpptraj.in
$ cd ../../
$ wget ↩
  ↪ https://raw.githubusercontent.com/pcyra2/MD_workshop/main/Files ↩
  ↪ /Dynamics/convert.in
$ cpptraj convert.in
```

6.g Task 7: Calculate the free energy of binding.

Command 6.7: Calculating the free energy.

bash command

```
$ cd ../Free_Energy/
$ cp ../Dynamics/prod.nc .
$ cp ../Dynamics/complex.parm7 SOLVATED_COMPLEX.parm7
$ ante-MMPBSA.py -p SOLVATED_COMPLEX.parm7 -c complex.parm7 -r
  ↪ receptor.parm7 -l ligand.parm7 -s :WAT,Na+ -n :UNL
  ↪ --radii=mbondi2
$ wget
  ↪ https://raw.githubusercontent.com/pcyra2/MD_workshop/main/Files
  ↪ /Energy/gbsa.in
$ mpirun -np 10 MMPBSA.py.MPI -O -i gbsa.in -o gb_results.dat -sp
  ↪ SOLVATED_COMPLEX.parm7 -cp complex.parm7 -rp receptor.parm7
  ↪ -lp ligand.parm7 -y prod.nc
$ tail gb_results.dat
```

7 Bibliography

References

1. J. W. Ponder, D. A. Case, in, pp. 27–85, DOI 10.1016/S0065-3233(03)66002-X.
2. K. Vanommeslaeghe *et al.*, CHARMM general force field: A force field for drug-like molecules compatible with the CHARMM all-atom additive biological force fields. *J. Comput. Chem.*, NA–NA, DOI 10.1002/jcc.21367 (2009).
3. J. A. Maier *et al.*, ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from ff99SB. *J. Chem. Theo. Comput.* **11**, 3696–3713, DOI 10.1021/acs.jctc.5b00255 (Aug. 2015).
4. P. A. Kollman *et al.*, Calculating Structures and Free Energies of Complex Molecules: Combining Molecular Mechanics and Continuum Models. *Accounts of Chemical Research* **33**, 889–897, DOI 10.1021/ar000033j (Dec. 2000).
5. M. Weitman, D. T. Major, Challenges Posed to Bornyl Diphosphate Synthase: Diverging Reaction Mechanisms in Monoterpenes. *Journal of the American Chemical Society* **132**, 6349–6360, DOI 10.1021/ja910134x, (https://pubs.acs.org/doi/10.1021/ja910134x) (May 2010).
6. William Humphrey, Andrew Dalke, Klaus Schulten, VMD - Visual Molecular Dynamics. *Journal of Molecular Graphics* **14**, 33–38 (1996).
7. A. T. McNutt *et al.*, GNINA 1.0: molecular docking with deep learning. *Journal of Cheminformatics* **13**, 1–20, DOI 10.1186/S13321-021-00522-2/FIGURES/13, (https://jcheminf.biomedcentral.com/articles/10.1186/s13321-021-00522-2) (Dec. 2021).
8. A. W. Sousa da Silva, W. F. Vranken, ACPYPE - AnteChamber PYthon Parser interfacE. *BMC Res. Notes* **5**, 367, DOI 10.1186/1756-0500-5-367 (Dec. 2012).
9. B. R. Brooks *et al.*, CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.* **4**, 187–217, (https://doi.org/10.1002/jcc.540040211) (1983).
10. H. J. Berendsen, D. van der Spoel, R. van Drunen, GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications* **91**, 43–56, DOI 10.1016/0010-4655(95)00042-E (Sept. 1995).

11. K. Vanommeslaeghe, A. D. MacKerell, Automation of the CHARMM general force field (CGenFF) I: Bond perception and atom typing. *Journal of Chemical Information and Modeling* **52**, 3144–3154, DOI 10.1021/CI300363C/SUPPL{_}FILE/CI300363C{_}SI{_}001.ZIP, (https://pubs.acs.org/doi/full/10.1021/ci300363c) (Dec. 2012).

12. G. M. Morris *et al.*, AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility. *Journal of computational chemistry* **30**, 2785, DOI 10.1002/JCC.21256, (/pmc/articles/PMC2760638/%20/pmc/articles/PMC2760638/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2760638/) (Dec. 2009).

13. E. Krieger, G. Vriend, YASARA View - molecular graphics for all devices - from smartphones to workstations. *Bioinformatics (Oxford, England)* **30**, 2981–2982, DOI 10.1093/BIOINFORMATICS/BTU426, (https://pubmed.ncbi.nlm.nih.gov/24996895/) (Oct. 2014).

14. J. Jumper *et al.*, Highly accurate protein structure prediction with AlphaFold. *Nat.* **596**, 583–589, DOI 10.1038/s41586-021-03819-2 (Aug. 2021).

15. M. Varadi *et al.*, AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Res.* **50**, D439–D444, DOI 10.1093/nar/gkab1061 (Jan. 2022).

16. A. Jakalian, D. B. Jack, C. I. Bayly, Fast, efficient generation of high-quality atomic charges. AM1-BCC model: II. Parameterization and validation. *Journal of computational chemistry* **23**, 1623–1641, DOI 10.1002/JCC.10128, (https://pubmed.ncbi.nlm.nih.gov/12395429/) (Dec. 2002).

17. J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, D. A. Case, Development and testing of a general amber force field. *J. Comput. Chem.* **25**, 1157–1174, DOI 10.1002/jcc.20035 (July 2004).

18. Z. Cournia, B. Allen, W. Sherman, Relative Binding Free Energy Calculations in Drug Discovery: Recent Advances and Practical Considerations. *J. Chem. Inf. Model.* **57**, 2911–2937, DOI 10.1021/acs.jcim.7b00564 (Dec. 2017).

19. S. Genheden, U. Ryde, Comparison of end-point continuum-solvation methods for the calculation of protein-ligand binding free energies. *Proteins: Struct., Funct., and Bioinf.* **80**, 1326–1342, DOI 10.1002/prot.24029 (May 2012).

20. N. Singh, A. Warshel, Absolute binding free energy calculations: On the accuracy of computational scoring of protein-ligand interactions. *Proteins: Struct., Funct., and Bioinf.*, NA–NA, DOI 10.1002/prot.22687 (2010).

21. A. P. Bhati, S. Wan, D. W. Wright, P. V. Coveney, Rapid, Accurate, Precise, and Reliable Relative Free Energy Prediction Using Ensemble Based Thermodynamic Integration. *J. Chem. Theo. Comput.* **13**, 210–222, DOI 10.1021/acs.jctc.6b00979 (Dec. 2016).

22. E. Wang *et al.*, End-Point Binding Free Energy Calculation with MM/PBSA and MM/GBSA: Strategies and Applications in Drug Design. *Chem. Rev*, **119**, 9478–9508, DOI 10.1021/acs.chemrev.9b00055 (Aug. 2019).

Acronyms

CLI command-line interface

MM/GBSA Molecular Mechanics/Generalized Born Surface Area

GUI graphical user interface

MD molecular dynamics

MM molecular mechanics

NVP Constant number of atoms, volume and pressure

NVT Constant number of atoms, volume and temperature

OS operating system

MM/PBSA Molecular Mechanics/Poisson–Boltzmann Surface Area

PS picoseconds

QM quantium mechanics

RMSD root mean square deviation

VDW van der Waals

WSL2 Windows subsystem for linux 2