Nisin PV; a bioengineered derivative resistant to the Nisin Resistance Protein.

Des Field[1,*], Tony Blake[3], Harsh Mathur[2], Paula M O Connor[2,3], Paul D Cotter[2,3], Paul Ross[1,3], & Colin Hill[1,3,**]

[1]*School of Microbiology, University College Cork, Cork, Ireland.*
[2]*Teagasc Food Research Centre, Moorepark, Fermoy, Co. Cork, Ireland.*
[3]*APC Microbiome Institute, University College Cork, Ireland.*

Molecular Modelling Workflow

The molecular modeling simulations consisted of 2 models. The first model used a configuration of residues 22-34 of a nisin mutant where residue 29 is serine and is structurally aligned so that residues 22-34 fit into the tunnel region of the NSR molecule. The second model was also aligned to fit into the tunnel region of the NSR enzyme only this time residue 29 has been mutated to proline.

Creating the NSR-nisin pdb file To create a pdb file for the NSR-Nisin complex individual pdb files for both NSR and nisin were required. This was carried out with relative ease using the R package bio3d like so [1].

```
> install.packages("bio3d")
> library(bio3d)
> # This code fetches the NSR pdb file and splits it into separate pdb files for each chain
> get.pdb("4Y68", URLonly = F, split=TRUE, path="split_chain.test", multi=F)
> # This code fetches the nisin file and spilt the pdb file into individual models
> # with a separate file for each chain
> get.pdb("1wco", URLonly = F, split=TRUE, path="split_chain.nisin", multi=T)
```

For the NSR molcule the pdb file `4Y68` was used and for the nisin the `1wco` pdb file was used where Protein Databank Accession numeber for both files came from their respective papers [2] [3].

Docking Procedure To determine a starting configuration for the NSR-Nisin complex a docking program Autodock for ligand and protein binding was used [4]. In the Using AutoDock 4 and AutoDock Vina with AutoDockTools:A Tutorial exercises 1-10 were folowed to carry out the docking procedure with 1wco.pdb as the ligand and 4Y68 as the receptor. This produced 9 possible binding conformations for the NSR-Nisin complex. Out of these 9 states the 7th conformation state was chosen being the state which showed most favorable interaction between residues 29 of nisin and residues 236-240 of the active site in NSR. In this model the NSR-nisin complex had serine for residue 29. The molecular visualisation package Chimera was then used to mutate residue 29 to proline and the subsequent configuration saved as a different pdb file. Again the docking program Autodock was used to create a starting configuration for the NSR-Nisin complex. In this case the 1st conformation was chosen for the same reasons as previously. Then both chosen conformation states (one with serine, the other with proline) were saved as separate pdb files to be used as starting configurations in the molcular modeling simulation.

Molecular Modelling To run the MD simulation the Amber workflow was used [5]. This consisted of the following steps.

- Prepare the pdb files using LEAP.
- Prepare cif files for nonstandrad residues DBB and DHA.
- Use antechamber forcefield values to fill in for missing parameters.
- Use antechamber to create prmtop and inpcrd files for each pdb file
- Use sander to run MD simulation from prmtop and inpcrd files
- Use cpptraj to analyse MD trajectory files for change of distances over time
- Use cpptraj to determine H-bonds and water mediated inteactions
- Use mmpbsa to calculate binding enrgies of residues

In what follows the procedure for workflow as applied to the NSR-Nisin model with serine is described. The steps for the workflow as applied to the NSR-Nisin model with proline are identical.

Here the Amber 16 suite of programs was used [5]. To start the pdb files from the docking procedure step were prepared for use with Amber's LEaP program [6]. This required running the following Linux command.

```
$ pdb4amber -i Nisinmol7.pdb -o gfp.pdb
$ pdb4amber -i 4Y68_A.pdb -o NSR.pdb
$ pdb4amber -i NSR.pdb -o recptor.pdb
$ reduce -Trim recptor.pdb > recptorNoH.pdb
```

Here Nisinmol7.pdb was the fie created by Autodock for the 7th binding conformation state and gfp.pdb the output file from the pdb4amber program. And `4Y68\_A.pdb` was the pdb file for chain A in the NSR protein. The `pdb4amber` program changed the residues labled HIS to HIE and indicated that the nonstandard residues dehydroalanine (DHA) and d-alpha-aminobutyric acid (DBB) were not recognised by LEaP. Also the reduce program with the -Trim flag strips all hydrogens from the pdb file [7]

To deal with the nonstandard residues dehydroalanine (DHA) and d-alpha-aminobutyric acid (DBB) the respective entries in the RCSB Protein Data Bank were accessed and the respective `.cif` files downloaded. The following BASH code was then run using several programs from Amber.

```
$ antechamber -fi ccif -i DBB.cif -bk DBB -fo ac -o dbb.ac -c bcc -at amber
$ prepgen -i dbb.ac -o dbb.prepin -m dbb.mc -rn DBB
$ parmchk2 -i dbb.prepin -f prepi -o dbb.frcmod -a Y -p parm10.dat
$ antechamber -fi ccif -i DHA.cif -bk DHA -fo ac -o dha.ac -c bcc -at amber
$ prepgen -i dha.ac -o dha.prepin -m dha.mc -rn DHA
$ parmchk2 -i dha.prepin -f prepi -o dha.frcmod -a Y -p parm10.dat
```

The amber program antechamber can read the `.cif` files of the nonstandard residues and assign partial charges and atom types to the the nonstandard residues based on the bcc charge scheme [8]. This will output .ac files which have charge and bonding information for the nonstandard residues. These will then used as input to the prepgen program along with a custom made mc file that tells the prepgen program what atoms to ignore from the residue (for peptide bonding). The mc file should look like so

```
HEAD_NAME N
TAIL_NAME C
MAIN_CHAIN CA
```

```
OMIT_NAME OXT
OMIT_NAME HXT
PRE_HEAD_TYPE C
POST_TAIL_TYPE N
CHARGE 0.0
```

The HEAD_NAME and TAIL_NAME lines identify the atoms that will connect to the previous and following amino acids, respectively. The MAIN_CHAIN lines list the atoms along the chain that connect the head and the tail atoms. The OMIT_NAME lines list the atoms in the nonstandard residue that should be removed from the final structure, as they are not present in the intact protein. The PRE_HEAD_TYPE and POST_TAIL_TYPE lines let prepgen know what atom types in the surrounding protein will be used for the covalent connection. The CHARGE line gives the total charge on the residue; prepgen will ensure that the charges of the "omitted" atoms are redistributed among the remaining atoms so that the total charge is correct (i.e., 0 in this case).The prepgen program then outputs `.prepin` files which are the inputed into the `parmchk2` program that create the `.frcmod` files using paramters from the `gaff.dat` and `parm10.dat` parameter files. Next the `.prepin` and `.frcmod` files were read into the LEaP program

```
Macintosh-109add6f31eb:111.ROUGH tonyblake$ tleap
-I: Adding /Users/tonyblake/amber16/dat/leap/prep to search path.
-I: Adding /Users/tonyblake/amber16/dat/leap/lib to search path.
-I: Adding /Users/tonyblake/amber16/dat/leap/parm to search path.
-I: Adding /Users/tonyblake/amber16/dat/leap/cmd to search path.

Welcome to LEaP!
(no leaprc in search path)
> source leaprc.protein.ff14SB       #reads in ff14SB forcefield parameters
> set default PBRadii mbondi3
> loadamberprep dbb.prepin           #reads in prep file for DBB
> loadamberparams dbb.frcmod         #reads in forcefield parameters for DBB
> loadamberprep dha.prepin           #reads in prep file for DHA
> loadamberparams dha.frcmod         #reads in forcefield parameters for DHA
> x=loadPDB gfp.pdb                  #reads in gfp.pdb
> saveamberparm x gfp.prmtop gfp.inpcrd  # creates topology and coordinate files
```

At this point however LEaP began to complain about missing parameters for bond lengths, bond angles and dihedral angles. The different values for these parameters (they are different fior each atom) are used with the amber force field to determine the energies for the NSR-Nisin complex [9].

$$
\begin{aligned}
E_{total} &= \sum_{bonds} k_b(r - r_0)^2 \\
&+ \sum_{angles} k_\theta(\theta - \theta_0)^2 \\
&+ \sum_{dihedrals} V_n(1 + \cos(n\phi - \gamma)) \\
&+ \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\epsilon R_{ij}} \right]
\end{aligned}
$$

The values for parameters k_b, r_0, k_θ, θ_0, γ, V_n, A_{ij}, B_{ij} are specified in the `.frcmod` and `.dat` files. So to overcome the "missing parameter" issue values from the `gaff2.dat` file were used as the values for the missing parameter indicated by LEaP. The underlying cause of the "missing parameter" issue was LEaP's inability to recognise the 3 peptide bonds and associated angles and dihedral angles of LYS-DBB, ALA-DBB, and VAL-DHA. Thus the `extra.frcmod` file was created to supply these missing values to LEaP.

```
Macintosh-109add6f31eb:003.SANDER tonyblake$ vim extra.frcmod

Remark line goes here
MASS
NT 14.010        0.530              same as n4
C  12.010        0.616              same as c
O  16.000        0.434              same as o
CT 12.010        0.878              same as c3
CD 12.010        0.360              same as c2
CM 12.010        0.360              same as c2
CX 12.010        0.878              same as c3
N  14.01         0.530

BOND
C -NT  255.5    1.5460            same as c -n4

ANGLE
O-C -NT   69.53     118.830
C -NT-H   44.63     111.120          same as c -n4-hn SOURCE3_SOURCE5
C -NT-CT  62.14     108.760          same as c -n4-c3        SOURCE5
CX-C -NT  64.28     112.260          same as c3-c -n4        SOURCE3
CD-C -N   86.65     124.990           same as c2-c2-nh            SOURCE3
C -NT-CD  63.556    112.580          same as c2-n4-c2

DIHE

CX-C -NT-H    1    1.025       180.000          -2.
O-C -NT-CT    4   10.000       180.000           2.
```

```
CX-C -NT-CT   9    1.400         0.000           3.
O-C -NT-H     1    2.500       180.000          -2.
C -NT-CD-CM   6    0.000       180.000           3.
CX-C -NT-CD   9    1.400         0.000           3.
O-C -NT-CD    4   10.000       180.000           2.
C -NT-CD-C    6    0.000        80.000           3.
CM-CD-C -N    4   26.600       180.000           2.
NT-CD-C -N    4   26.600       180.000           2.

IMPROPER

NONBON
```

There was one other small issue which seems to be a bug with the LEaP program. In some instances after the extra.frcmod was read into leap and the command run for creating the topology and coordinate files LEaP would still not recognise the parameters for 1 or 2 dihedral angles

```
> loadamberparams extra.frcmod
Loading parameters: ./extra.frcmod
Reading force field modification type file (frcmod)
Reading title:
Remark line goes here
> saveamberparm x gfp.prmtop gfp.prmcrd
Checking Unit.
WARNING: The unperturbed charge of the unit: 2.000000 is not zero.

 -- ignoring the warning.

Building topology.
Building atom parameters.
Building bond parameters.
Building angle parameters.
Building proper torsion parameters.
 ** No torsion terms for  O-C-NT-CD
 ** No torsion terms for  CX-C-NT-CD
Building improper torsion parameters.
old PREP-specified impropers:
 <DBB 2>:  CA   +M   C    O
 <DBB 4>:  CA   +M   C    O
 <DHA 12>:  C    CB   CA   N
 <DHA 12>:  CA   HB1  CB   HB2
 <DHA 12>:  CA   +M   C    O
 total 30 improper torsions applied
 5 improper torsions in old prep form
Building H-Bond parameters.
Incorporating Non-Bonded adjustments.
Parameter file was not saved.
```

To solve this issue one only needs to move the the lines in the extra.frcmod corresponding to those 2 dihedral angles with missing parameters to the top of the dihedral angle parameter list. So if the above example had to be changed then the corrected version would look like so.

```
DIHE

O-C -NT-CD    4   10.000       180.000           2.
CX-C -NT-CD   9    1.400         0.000           3.
C -NT-CD-C    6    0.000        80.000           3.
CX-C -NT-H    1    1.025       180.000          -2.
O-C -NT-CT    4   10.000       180.000           2.
CX-C -NT-CT   9    1.400         0.000           3.
O-C -NT-H     1    2.500       180.000          -2.
C -NT-CD-CM   6    0.000       180.000           3.
CM-CD-C -N    4   26.600       180.000           2.
NT-CD-C -N    4   26.600       180.000           2.

IMPROPER

NONBON
```

Then LEaP is able to read all of the nisin molecule (gfp.pdb) and will create the corresponding topology (`gfp.prmtop`) and coordinate files (`gfp.prmcrd`). When this happens LEaP will output the following onto the screen

```
> loadamberparams extra.frcmod
Loading parameters: ./extra.frcmod
Reading force field modification type file (frcmod)
Reading title:
Remark line goes here
> saveamberparm x gfp.prmtop gfp.prmcrd
Checking Unit.
WARNING: The unperturbed charge of the unit: 2.000000 is not zero.

 -- ignoring the warning.

Building topology.
Building atom parameters.
Building bond parameters.
Building angle parameters.
Building proper torsion parameters.
Building improper torsion parameters.
old PREP-specified impropers:
 <DBB 2>:  CA   +M   C    O
 <DBB 4>:  CA   +M   C    O
 <DHA 12>:  C    CB   CA   N
```

```
 <DHA 12>:  CA   HB1  CB   HB2
 <DHA 12>:  CA   +M   C    O
 total 30 improper torsions applied
 5 improper torsions in old prep form
Building H-Bond parameters.
Incorporating Non-Bonded adjustments.
Not Marking per-residue atom chain types.
Marking per-residue atom chain types.
  (Residues lacking connect0/connect1 -
   these dont have chain types marked:

res total affected

CLYS 1
NLYS 1
  )
 (no restraints)
```

Then to carry out the rest of the LEaP procedure for the NSR enzyme, solvating nisin and NSR, combining nisin and NSR, and adding counterions the following commands were issued.

```
> source leaprc.water.tip3p
> solvateBox x TIP3PBOX 10.0
> saveamberparm x gfp.wat.prmtop gfp.wat.prmcrd
> REC=loadPDB recptor.pdb
> NSR=loadPDB recptorNoH.pdb
> saveamberparm NSR nsr.prmtop nsr.prmcrd
> solvateBox NSR TIP3PBOX 10.0
> saveamberparm NSR nsr.wat.prmtop nsr.wat.prmcrd
> LIG=loadPDB gfp.pdb
> saveamberparm LIG test.prmtop test.prmcrd
> PROT=loadPDB recptorNoH.pdb
> saveamberparm PROT test2.prot.prmtop test2.prot.prmcrd
> COM = combine {PROT LIG}
> saveamberparm COM com.prmtop com.prmcrd
> solvateBox COM TIP3PBOX 10.0
> saveamberparm COM com.wat.prmtop com.wat.prmcrd
> addIons x Cl- 0
> saveamberparm x gfp.neutral.prmtop gfp.neutral.prmcrd
> addIons LIG Cl- 0
> saveamberparm LIG gfp.neutral2.prmtop gfp.neutral2.prmcrd
> saveamberparm x gfp.wat.neutral.prmtop gfp.wat.neutral.prmcrd
> addIons PROT Cl- 0
> saveamberparm PROT nsr.neutral2.prmtop nsr.neutral2.prmcrd
> addIons COM Cl- 0
> saveamberparm COM com.neutral2.prmtop com.neutral2.prmcrd
> saveamberparm COM com.wat.neutral2.prmtop com.wat.neutral2.prmcrd
```

```
> addIons NSR Cl- 0
> saveamberparm NSR nsr.wat.neutral2.prmtop nsr.wat.neutral2.prmcrd
> LIGNEUT=loadPDB gfp.pdb
> saveamberparm LIGNEUT test3.prmtop test3.prmcrd
> PROTNEUT=loadPDB recptorNoH.pdb
> COMNEUT=combine {PROTNEUT LIGNEUT}
> addIons COMNEUT Cl- 0
> save COMNEUT com.neutral0.prmtop com.neutral0.prmcrd
> saveamberparm COMNEUT com.neutral0.prmtop com.neutral0.prmcrd
```

These commands produced many files that needed to be used with *sander*, *cpptraj* and *MMPBSA.py*. Next the sander program was used to carry out the molecular simulation. To use this program an input file needed to be created that would inform the program of the physical processes to simulate. Noting the good results of a previous study [2], the same parameters were employed. Firstly the system was minimisied in a two stage process. The input file for the first minimisation stage looks like this.

```
min.in:
&cntrl
imin=1, maxcyc=250, ncyc=50, ntmin=1,ntx = 1, ntc = 1, ntf = 1, ntb = 1,
ntp = 0, ntwx = 10000, ntwe = 0, ntpr = 10000, cut = 8.0,
ntr = 1, restraintmask = ':1-287 & !@H=', restraint_wt = 25.0, /
```

This means that harmonic restraints with a force constant of 25 kcal.mol^{-1}Å^{-2} were applied to all protein atoms while all other atoms were free to move during 50 cycles of steepest descent (SD) and 200 cycles of conjugate gradient (CG) minimization. In the second stage (using a separate input file with adjusted parameters, min2.in), the force constant of the harmonic restraints was reduced to 5 kcal.mol^{-1}Å^{-2}, and 50 cycles of SD and 200 cycles of CG minimization were performed.

```
sander -O -i min.in -o min.out -p com.wat.neutral2.prmtop -c com.wat.neutral2.prmcrd /
-r 01\_Min.rst -inf 01\_Min.mdinfo -ref com.wat.neutral2.prmcrd
```

The sander program takes the input file min.in with the instructions on how to minimise, the topology file `com.wat.neutral2.prmtop` also as input, the coordinate file `com.wat.neutral2.prmcrd` again as input and the cooridinate file to act as reference for the restraint intsructions. It outputs the `min.out` file contain information on each time step of the minisation phase and a restart file `01_Min.rst` so the command for the second stage knows the coordinates to start from.

```
sander -O -i min.in -o min.out -p com.wat.neutral2.prmtop -c com.wat.neutral2.prmcrd /
-r 01\_Min.rst -inf 01\_Min.mdinfo -ref com.wat.neutral2.prmcrd

sander -O -i min2.in -o min2.out -p com.wat.neutral2.prmtop -c 01\_Min.rst /
-r 01\_Min2.rst -inf 01\_Min.mdinfo -ref 01\_Min.rst
```

Next the system is heated up for a period of 50 picoseconds from 100K ro 300K with volume held constant. This is the input file for the heating stage.

```
heat.in:
&cntrl
nstlim=25000, dt=0.002, ntx=1, irest=0, ntpr=500, ntwr=500, ntwx=500,
tempi =100.0, temp0=300.0, ntt=1, tautp=2.0, ig=-1, ntc = 2, ntf = 2,
ntp = 0, ntb=1, nrespa=2, ntwe = 0, cut = 8.0,
ntr = 1, restraintmask = ':1-287 & !@H=', restraint_wt = 5.0, /
```

Then the density was adjusted to 1 gcm^{-3} during 30 ps while keeping pressure constant . The input file for this stage looked like this

```
equib.in:
&cntrl
nstlim=15000, dt=0.002, ntx=5, irest=1, ntpr=500, ntwr=500, ntwx=500, temp0=300.0,
ntt=1, tautp=2.0, ig=-1, ntc = 2, ntf = 2, ntp = 1, ntb=2, nrespa=1, ntwe = 0, cut = 8.0,
ntr = 1, restraintmask = ':1-287 & !@H=', restraint_wt = 5.0, /
```

Next the positional restraints were gradually reduced from 5 $\text{kcal.mol}^{-1}\text{Å}^{-2}$ to 0 $\text{kcal.mol}^{-1}\text{Å}^{-2}$ while keeping the volume constant. This was achieved in MD simulation by a series of 6 stages (each stage having a different input file with the value for `restraint_wt` being going from 5 to 0). Each stage was 10 picoseconds long. As an example here's the input file for when the restraint was 3 $\text{kcal.mol}^{-1}\text{Å}^{-2}$

```
equib4.in:heat com.neutral0
&cntrl
nstlim=5000, dt=0.002, ntx=5, irest=1,
ntpr=500, ntwr=500, ntwx=500, temp0=300.0,
ntt=1, tautp=2.0, ig=-1, ntc = 2, ntf = 2, ntp = 0,
ntb=1, nrespa=2, ntwe = 0, cut = 8.0,
ntr = 1, restraintmask = ':1-287 & !@H=', restraint_wt = 3.0, /
```

The commands to run these processes look like so. Note also the .mdcrd files. These are the most important files (trajectory files) as they contain the binary informtaion that the programs *cpptraj* and *MMPBSA.py* will make use of to analyse the MD trajectories.

```
$ sander -O -i heat.in -o heat.out -p com.wat.neutral2.prmtop -c 01\_Min2.rst /
-r heat.rst -ref 01\_Min2.rst -x heat.mdcrd
$ sander -O -i equib.in -o equib.out -p com.wat.neutral2.prmtop -c heat.rst /
-r equib.rst -ref heat.rst -x equib.mdcrd
$ sander -O -i equib2.in -o equib2.out -p com.wat.neutral2.prmtop -c equib.rst /
-r equib2.rst -ref equib.rst -x equib2.mdcrd
$ sander -O -i equib3.in -o equib3.out -p com.wat.neutral2.prmtop -c equib2.rst /
-r equib3.rst -ref equib2.rst -x equib3.mdcrd
$ sander -O -i equib4.in -o equib4.out -p com.wat.neutral2.prmtop -c equib3.rst /
-r equib4.rst -ref equib3.rst -x equib4.mdcrd
$ sander -O -i equib5.in -o equib5.out -p com.wat.neutral2.prmtop -c equib4.rst /
-r equib5.rst -ref equib4.rst -x equib5.mdcrd
sander -O -i equib6.in -o equib6.out -p com.wat.neutral2.prmtop -c equib5.rst /
$ -r equib6.rst -ref equib5.rst -x equib6.mdcrd
sander -O -i equib6.in -o equib6.out -p com.wat.neutral2.prmtop -c equib5.rst /
-r equib7.rst -ref equib6.rst -x equib7.mdcrd
```

Finally the input file was changed to instruct a production run of 50 nanoseconds. This meant setting the total number of timesteps to =25000000. The command to run this part of the simulation uses the parallel processes program MPI (as otherwise it would take several years to complete the simulation).

```
$ mpirun -np 4 $AMBERHOME/bin/sander.MPI -O -i prodser1ns1.in -p com.wat.neutral2.prmtop
-c prod40.rst -r prod41.rst -o prod41.out -x prod41.mdcrd
```

For practical purposes the production run was split up into many stages (so as to able recover completed trajectory files in the event of a crash and also to not ”hog the server”). So the previous line of code shows very near to the end of all the timesteps (from 33 to 34 nanoseconds). Througout the full MD simulation long-range electrostatic interactions were treated using the particle mesh Ewald method[ref]. A distance cut off of 8 Angstroms was used to define short range electrostatic interactions. All bonds involving hydrogen were constrained by using the SHAKE algorithim [ref] which also required setting the timestep to be 2 femtoseconds. Trajectory files were created every picosecond.

Trajectory Analysis To analyse the MD trajectory the programs *cpptraj* and *MMPBSA.py* were used. To calculate the distance between the carbonyl carbon of CYS28 (NSR cleave point in nisin)

and the sidechain Oxygen in residue 236 of NSR (active site) the dist command in cpptraj was used. The only requirments for *cpptraj* were the topology file com.wat.neutral2.prmtop and an input file specifying what trajectory files were to be used. This is an example of the cpptraj command used and also the contents of the input file.

```
$ cpptraj -p com.wat.neutral2.prmtop -i trajfiles.atom.ptraj

# input file trajfiles.atom.ptraj
trajin heat.mdcrd
trajin equib1.mdcrd
trajin equib2.mdcrd
trajin equib3.mdcrd
trajin equib4.mdcrd
trajin equib5.mdcrd
trajin equib6.mdcrd
trajin equib7.mdcrd
trajin test2.mdcrd
trajin prod2.mdcrd
trajin prod3.mdcrd
trajin prod4.mdcrd
trajin prod5.mdcrd
trajin prod6.mdcrd
trajin prod7.mdcrd
trajin prod8.mdcrd
trajin prod9.mdcrd
trajin prod10.mdcrd
trajin prod11.mdcrd
trajin prod12.mdcrd
trajin prod13.mdcrd
trajin prod14.mdcrd
trajin prod15.mdcrd
trajin prod16.mdcrd
trajin prod17.mdcrd
trajin prod18.mdcrd
trajin prod19.mdcrd
trajin prod20.mdcrd
trajin prod21.mdcrd
trajin prod22.mdcrd
trajin prod23.mdcrd
trajin prod24.mdcrd
trajin prod25.mdcrd
trajin prod26.mdcrd
trajin prod27.mdcrd
trajin prod28.mdcrd
trajin prod29.mdcrd
trajin prod30.mdcrd
trajin prod31.mdcrd
trajin prod32.mdcrd
```

```
trajin prod33.mdcrd
trajin prod34.mdcrd
trajin prod35.mdcrd


distance SER236OtoCYS28C :206@3402 :294@4762 out SER236OtoCYS28C

run
```

After the trajectory analysis has finished the output file SER236OtoCYS28C contains the distance between both points of interest for every picosecond of the trajectory. These can then be plotted using the ggplot2 program in R. The next part of the trajectory analysis involves determining what hydrogen bonds have formed over the duration of the simulation (35 nanoseconds). This was achieved by again using the *cpptraj* program and a different input file. Here is an example of the code and input file.

```
$ cpptraj -p com.wat.neutral2.prmtop -i hbondtraj.ptraj

# input file hbondtraj.ptraj
trajin heat.mdcrd
trajin equib1.mdcrd
trajin equib2.mdcrd
trajin equib3.mdcrd
trajin equib4.mdcrd
trajin equib5.mdcrd
trajin equib6.mdcrd
trajin equib7.mdcrd
trajin test2.mdcrd
trajin prod2.mdcrd
trajin prod3.mdcrd
trajin prod4.mdcrd
trajin prod5.mdcrd
trajin prod6.mdcrd
trajin prod7.mdcrd
trajin prod8.mdcrd
trajin prod9.mdcrd
trajin prod10.mdcrd
trajin prod11.mdcrd
trajin prod12.mdcrd
trajin prod13.mdcrd
trajin prod14.mdcrd
trajin prod15.mdcrd
trajin prod16.mdcrd
trajin prod17.mdcrd
trajin prod18.mdcrd
```

```
trajin prod19.mdcrd
trajin prod20.mdcrd
trajin prod21.mdcrd
trajin prod22.mdcrd
trajin prod23.mdcrd
trajin prod24.mdcrd
trajin prod25.mdcrd
trajin prod26.mdcrd
trajin prod27.mdcrd
trajin prod28.mdcrd
trajin prod29.mdcrd
trajin prod30.mdcrd
trajin prod31.mdcrd
trajin prod32.mdcrd
trajin prod33.mdcrd
trajin prod34.mdcrd
trajin prod35.mdcrd


hbond All out All.hbvtime.dat solventdonor :WAT solventacceptor :WAT@O \
  avgout All.UU.avg.dat solvout All.UV.avg.dat bridgeout All.bridge.avg.dat

hbond Backbone :288-299@C,O,N,H avgout BB.avg.dat series uuseries bbhbond

create nhbvtime All[UU] Backbone[UU] All[UV] All[Bridge]

rms BBrmsd :288-299@C,CA,N out BBrmsd

run
```

The important file to note here is the All.UU.avg.dat file. This contains all the percentages for hydrogen bond formation over the course of the trajectory. After the hydrogen bond calculation the *MMPBSA.py* was used to calculate the binding energies of the residues in nisin and the residues in the active site of (TASSAEM motif) NSR. This also required many of the files prepared earlier in the analysis using the LEaP program. Again here is an example of the command and the contents of the input file.

```
$AMBERHOME/bin/MMPBSA.py -O -i mmpbsa_per_res_decomp.in /
-o FINAL_RESULTS_MMPBSA.dat -do FINAL_DECOMP_MMPBSA.dat /
-sp com.wat.neutral2.prmtop -cp com.prmtop -rp nsr.prmtop -lp gfp.prmtop -y *.mdcrd

# input file mmpbsa_per_res_decomp.in
Per-residue GB and PB decomposition
&general
   endframe=35000, verbose=1, interval=100
```

```
/
&gb
  igb=5, saltcon=0.100,
/
&decomp
  idecomp=1, print_res="288-300"
  dec_verbose=1,
/
```

One final point to note. For the binding enbergy calculations the MM-GBSA (Generalised Born) method was used and not the MM-PBSA (Poisson-Boltzmann) method. The FINAL_DECOMP_MMPBSA.dat file has all the values of the binding energies for the residues specified in the input file and can then plotted using the ggplot2 in R [10,11].

Lastly, the program to extract the RMSD and RMSF information from the trajectory files is almost the same as that used for the distance information extraction.

```
```bash
$ cpptraj -p com.wat.neutral2.prmtop -i trajfiles.rmstime50.ptraj

# Contents of input file trajfiles.rmstime50.ptraj
trajin heat.mdcrd
trajin equib1.mdcrd
trajin equib2.mdcrd
trajin equib3.mdcrd
trajin equib4.mdcrd
trajin equib5.mdcrd
trajin equib6.mdcrd
trajin equib7.mdcrd
trajin test2.mdcrd
trajin prod2.mdcrd
trajin prod3.mdcrd
trajin prod4.mdcrd
trajin prod5.mdcrd
trajin prod6.mdcrd
trajin prod7.mdcrd
trajin prod8.mdcrd
trajin prod9.mdcrd
trajin prod10.mdcrd
trajin prod11.mdcrd
trajin prod12.mdcrd
trajin prod13.mdcrd
trajin prod14.mdcrd
trajin prod15.mdcrd
trajin prod16.mdcrd
```
```

```
trajin prod17.mdcrd
trajin prod18.mdcrd
trajin prod19.mdcrd
trajin prod20.mdcrd
trajin prod21.mdcrd
trajin prod22.mdcrd
trajin prod23.mdcrd
trajin prod24.mdcrd
trajin prod25.mdcrd
trajin prod26.mdcrd
trajin prod27.mdcrd
trajin prod28.mdcrd
trajin prod29.mdcrd
trajin prod30.mdcrd
trajin prod31.mdcrd
trajin prod32.mdcrd
trajin prod33.mdcrd
trajin prod34.mdcrd
trajin prod35.mdcrd
trajin prod36.mdcrd
trajin prod37.mdcrd
trajin prod38.mdcrd
trajin prod39.mdcrd
trajin prod40.mdcrd
trajin prod41.mdcrd
trajin prod42.mdcrd
trajin prod43.mdcrd
trajin prod44.mdcrd
trajin prod45.mdcrd
trajin prod46.mdcrd
trajin prod47.mdcrd
trajin prod48.mdcrd
trajin prod49.mdcrd

rms ToFirst :288-300&!@H= first out rmsdnisinovertime mass

run
```

After running the above commands “‘trajfiles.rmstime50.ptraj“‘ several files are created. The file “‘rmsdnisinovertime“‘ shows the values of the atomic positional fluctuations (RMSF) calculated for all atoms in the mask of residues 22 to 34 Nisin at every picosecond. And similarly for the RMSD calculation.

```
$ cpptraj -p com.wat.neutral2.prmtop -i trajfiles.rmsd50.ptraj

# input file trajfiles.rmsd50.ptraj
```

```
trajin heat.mdcrd
trajin equib1.mdcrd
trajin equib2.mdcrd
trajin equib3.mdcrd
trajin equib4.mdcrd
trajin equib5.mdcrd
trajin equib6.mdcrd
trajin equib7.mdcrd
trajin test2.mdcrd
trajin prod2.mdcrd
trajin prod3.mdcrd
trajin prod4.mdcrd
trajin prod5.mdcrd
trajin prod6.mdcrd
trajin prod7.mdcrd
trajin prod8.mdcrd
trajin prod9.mdcrd
trajin prod10.mdcrd
trajin prod11.mdcrd
trajin prod12.mdcrd
trajin prod13.mdcrd
trajin prod14.mdcrd
trajin prod15.mdcrd
trajin prod16.mdcrd
trajin prod17.mdcrd
trajin prod18.mdcrd
trajin prod19.mdcrd
trajin prod20.mdcrd
trajin prod21.mdcrd
trajin prod22.mdcrd
trajin prod23.mdcrd
trajin prod24.mdcrd
trajin prod25.mdcrd
trajin prod26.mdcrd
trajin prod27.mdcrd
trajin prod28.mdcrd
trajin prod29.mdcrd
trajin prod30.mdcrd
trajin prod31.mdcrd
trajin prod32.mdcrd
trajin prod33.mdcrd
trajin prod34.mdcrd
trajin prod35.mdcrd
trajin prod36.mdcrd
trajin prod37.mdcrd
trajin prod38.mdcrd
trajin prod39.mdcrd
trajin prod40.mdcrd
trajin prod41.mdcrd
```

```
trajin prod42.mdcrd
trajin prod43.mdcrd
trajin prod44.mdcrd
trajin prod45.mdcrd
trajin prod46.mdcrd
trajin prod47.mdcrd
trajin prod48.mdcrd
trajin prod49.mdcrd

rmsd nisincoreBBcheck :288-300@C,CA,N perres perresout rms_vs_time.BB.50.dat perresavg perresavg.BB.50.dat

run
```

After running the same commands for `trajfiles.rmsd50.ptraj` two files are created. One for the RMSD over time `rms_vs_time.BB.50.dat` and one for the average-over-time RMSD per residue `perresavg.BB.50.dat`. For the purposes of plotting in R a bar chart was made using the data in the `perresavg.BB.50.dat` file.

References

1. Grant, B. J., Rodrigues, A. P. C., ElSawy, K. M., McCammon, J. A. & Caves, L. S. D. Bio3d: an r package for the comparative analysis of protein structures. *Bioinformatics* **22**, 2695 (2006). URL + http://dx.doi.org/10.1093/bioinformatics/btl461. /oup/backfile/Content$_public/Journal/bioinformatics/22/21/10.1093/bioinformatics/btl461/2/btl$

2. Khosa, S. *et al.* Structural basis of lantibiotic recognition by the nisin resistance protein from streptococcus agalactiae. *Sci. Rep.* **6**, 18679 ((2016)).

3. Hsu, S. *et al.* The nisin-lipid ii complex reveals a pyrophosphate cage that provides a blueprint for novel antibiotics. *Nat Struct Mol Biol* **11**, 963 (2004).

4. van Nuland, N. A. J. *et al.* Autodock4 and autodocktools4: Automated docking with selective receptor flexibility. *J Comput Chem* **30**, 2785–2791 (2009).

5. Case, D. *et al.* Amber16. *University of California, San Francisco* .

6. Zhang, T., W.and Hou, Schafmeister, C., Ross, W. & Case, D. Leap. *University of California, San Francisco* (1995).

7. Word *et al.* Leap. *J. Mol. Biol.* **285**, 1735–1747 (1999).

8. Jakalian, A., Jack, D. B. & Bayly, C. I. Fast, efficient generation of high-quality atomic charges. am1-bcc model: Ii. parameterization and validation. *Journal of Computational Chemistry* **23**, 1623–1641 (2002). URL http://dx.doi.org/10.1002/jcc.10128.

9. Cornell, W. D. *et al.* A second generation force field for the simulation of proteins, nucleic acids, and organic molecules j. am. chem. soc. 1995, 117, 51795197. *Journal of the American Chemical Society* **118**, 2309–2309 (1996). URL http://dx.doi.org/10.1021/ja955032e. http://dx.doi.org/10.1021/ja955032e.

10. R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org.

11. Wickham, H. *ggplot2: Elegant Graphics for Data Analysis* (Springer-Verlag New York, 2009). URL http://ggplot2.org.