NAMD Tutorial

Introduction and the Basics

Nanoscale Molecular Dynamics/Not Another Molecular Dynamics program (NAMD) is a molecular dynamics (MD) program designed for scaling with significant amount of computational resources – this makes it particularly useful for very large molecules. In addition to this, NAMD is designed with biomolecules in mind and the force fields available for use in NAMD have large libraries for use in simulations of these biomolecules. A particular advantage of NAMD is easy access to a CUDA-enabled version, allowing the use of GPUs to dramatically accelerate the rate of molecular dynamics simulations – the scalability with GPUs is such that simulations with over 240 million individual atoms were performed with good performance on over 16,000 nodes. Another advantage it shares with several competitors is price – it is a free program that can be downloaded and installed by anyone. Competitors such as GROMACS and LAMMPS are also free, while Amber is the key competitor that is not free. Amber is a powerful MD package that is designed for exclusive use with Amber force fields, GROMACS is a more versatile MD package that is focused on speed of calculation and is more commonly used for non-biomolecular MD simulations than the above packages, and finally LAMMPS is a very modular, versatile package focused on a wide range of options. As the rest of this tutorial deals with NAMD, hopefully its advantages coincide with what you want!

The NAMD MD package supports several possible force fields – the set of parameters that control how atoms interact in MD. The primary force fields available in NAMD are CHARMM, X-PLOR (a modification of CHARMM designed to facilitate interaction with experimental data), GROMOS, and Amber. Of these, CHARMM is the most commonly used with NAMD – as Amber has its own dedicated MD package, and GROMOS is easier to use in GROMACS. The Amber force fields are primarily focused on protein biopolymers but do also have parameters available for carbohydrates and lipids – and unlike the MD package, the force fields are freely available. GROMOS force fields are united-atom (UA) force fields – aliphatic carbon atoms have the mass of directly bonded hydrogen atoms added to them, and the associated hydrogen atoms are not explicitly modelled. The CHARMM/X-PLOR force fields are more diverse, with united-atom or all-atom variants available, and extensively developed parameters for lipids, nucleotide bases, carbohydrates, proteins, and recently small organic molecules. This tutorial will assume the use of CHARMM parameters – but the process is similar for any of the above.

Before any simulation can be run, NAMD, VMD (Visual Molecular Dynamics; a program to visualize and analyze MD results), your chosen force field, and your starting structure must all be available to you. NAMD and VMD can be downloaded from the Theoretical and Computational Biophysics Group of the University of Illinois free of charge, currently at the following addresses: https://www.ks.uiuc.edu/Research/namd/ and https://www.ks.uiuc.edu/Research/vmd/. The various forcefields are available from the following sites:

Forcefield	Website	Recommended Version
CHARMM	http://mackerell.umaryland.edu/charmm_ff.shtml	CHARMM36[1]/22*[2]
GROMOS	http://www.gromos.net/	GROMOS 54a8[3]
Amber	http://ambermd.org/AmberModels.php	ff14SB[4]/ff15ipq[5]/ff15FB[6]

(for more detailed information on recommended versions, see extra information or the listed references)

The starting structures used in simulations are obtained from several potential sources – the source will vary depending on the structures being studied. The Protein Data Bank contains crystal structures and NMR solution structures of every protein that has been crystallized and is publicly available, as well as many DNA and RNA structures. Structures of lipids and carbohydrates are more often procedurally generated or created in programs like Avogadro and converted to the .pdb format. All structures being input into NAMD must be in the .pdb file format. A helpful site for the generation of inputs into NAMD is the CHARMM-GUI website – it is capable of skipping the setup phase for many calculations, and also has useful tools such as the generation of lipid bilayers. It is possible to use this to skip a good portion of the initial setup required for MD simulations, but I would recommend avoiding this initially - MD does require quite a great deal of understanding of these stages to ensure that they're done correctly, and it is best to have some experience doing these tasks before automating them.

Running a Simulation

First, we'll run through the steps needed to run a simulation, and when that's completed, we'll work through an example. The first step in running a simulation is ensuring you have access to VMD, NAMD, and your chosen force field (CHARMM36 in the case of the example). The links to the download of these programs and packages are available above, and installation instructions are included (and vary depending on the machine you are installing it on). Once you have ensured these programs are installed and available, the next step is to get access to the starting structure in a .pdb format; this format contains information on the location of atoms relative to each other, as well as the element of the atom and the chain of an atom (with each chain being a series of covalently-bonded monomers). This is of great importance, but it is not all that is required – the connectivity of the atoms is also required. A Protein Structure File (.psf) can be generated from the positioning of the atoms (the .pdb file you start with) and a special type of parameter (known as the topology) that details the interconnectivity of the parameterized monomers. The program that does this is bundled with NAMD and is named psfgen – an input file is required that details the location of your topology files, the .pdb file, and lists each chain in the input .pdb file. Further operations can be done at this point if appropriate – modifications to the final state (known as patches) can be applied that change partial charges, connectivity, and even delete or add atoms. A common example of this is the addition of a disulfide bridge – deleting the hydrogen atoms attached to two (spatially adjacent) thiol groups and forming a bond between the two, or the conversion of the first and last residues of each chain to capped residues which is done automatically for recognized amino acids.

With the .psf file generated, a complete description of the system is available – the coordinates of the atoms are stored in the .pdb file, and how they are connected is stored in the .psf file. The next stop in almost all calculations is the solvation and ionization of the biomolecule you have represented. This can easily be done using VMD in one of two ways; an extension is in-built to allow solvation using a GUI, or a VMD script can be written in the TCL scripting language that solvates and ionizes a system. Adding ions to the solution (referred to as ionization) can be done with a variety of cations – sodium, magnesium, potassium, cesium, calcium, and zinc – but chloride is the only available anion. Ionization

can be used to neutralize the system (which is a requisite of most modern MD methods), or to neutralize the system and continue adding salt until a set salt concentration is reached. In both cases, water is the default solvent, but any other solvent can be used provided you have access to an equilibrated box of the solvent from which to randomly sample; this box defaults to a rectangular or square shape (see Extra Information).

Once you have solvated and ionized your system, it is essentially ready for simulation to begin. A visual inspection (using VMD) is recommended to ensure nothing abnormal has occurred up until this point, but once this is completed a configuration file for the upcoming simulation is all that is needed. Configuration files detail what is to actually occur in a simulation – the temperature at which it occurs, how long each timestep is, how often it will write its progress, what parameters to be using, how to handle non-bonded forces, and so on. A full description of available options in a configuration file is available in the NAMD User's Guide, but a set of sample configuration files (for different types of simulations) are supplied alongside this tutorial (see Extra Information). In addition, a table in Extra Information details many of the commonly used options in a configuration file. These files tend to be long, but imminently reusable between similar simulations – my advice would be to make a fairly detailed configuration file initially, and then modify it from there for each individual simulation. If you're a more visual learner, a flowchart of the steps laid out here is available in Extra Information.

With a configuration file completed, you're ready to start a simulation – you simply need to run NAMD with the configuration file as the argument. But most of your jobs aren't going to be run locally – they'll be run on supercomputers, where there are some extra steps that need to be resolved. MD scales well with increased core count, particularly when using NAMD, but does require a large system to scale effectively; the simulation box is cut into patches that are roughly the size of the long-range interaction cutoff, and each patch is split into (roughly 14) compute objects. Each compute object can be assigned to a separate processor – and so the larger your system, the more patches and more compute objects, and so the more parallelizable your system is. NAMD runs benchmarking automatically at the start of each simulation, printing the number of days required to complete a nanosecond of simulation as well as the fraction of a second taken per step; if you are interested in the optimal number of cores to apply to a simulation, you can run 1-hour jobs and compare the benchmarking. NAMD doesn't rely on an MPI framework for the underlying parallelization, and so any number of cores can work – for a simulation under 50,000 atoms, I would not normally request more than a single node.

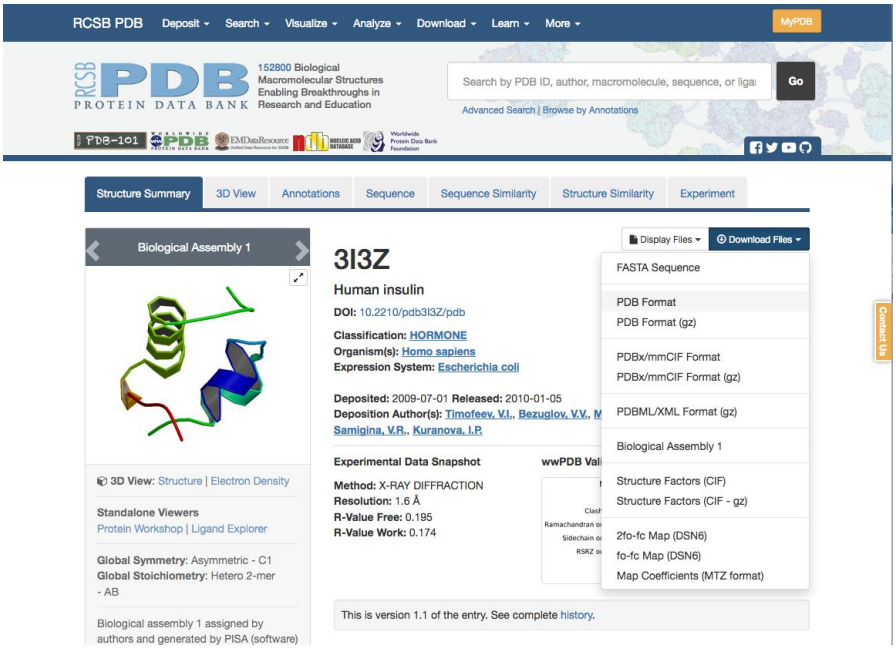
The number of cores used is impacted by the availability of GPUs on the supercomputer being used – NAMD simulations can be dramatically accelerated by the use of CUDA GPUs (almost any modern NVIDIA GPU); for example, an 11,000-atom simulation of mine was shortened from 12 days to 2 days when adding 2 P100 NVIDIA GPUs. These GPUs do require a sufficient number of atoms and CPUs to be worthwhile – typically needing around 10,000 atoms per GPU as well as somewhere between 8 and 25 CPUs per GPU. Given the P100 nodes on our local supercomputers have 28 CPUs for 2 GPUs, a 14:1 ratio is a good choice for moderately sized systems. As many algorithms in the computational field continue to struggle with GPU acceleration, I would recommend taking advantage of the shorter queues and faster speeds available through GPUs if they are available. When submitting job files that run NAMD, no extra command is required to handle GPU support (provided you are

running a CUDA-compiled version of NAMD), as the GPUs are automatically recognized - and in fact are required for CUDA-accelerated versions.

The final bit of more general advice before moving on to an example is about restart files; upon successfully completing a simulation, NAMD will output a .coor file (equivalent to a .pdb; it contains the coordinates of the system), a .vel file (containing the velocities of all atoms in the system), and finally a .xsc file (containing records of the periodic cell) in a binary format (by default); if the simulation crashes or times out, the latest version of these restart files will be present in the output folder (with the .restart suffix). These can be used to resume simulation, and simply require the next simulation to use the binCoordinates and binVelocities commands instead of the coordinates and temperature commands, respectively. For equilibration simulations where one might run an NVT simulation, followed by an NPT simulation, followed by a simulation in which restraints are relaxed, I would recommend submitting these short simulations together as one job, with each configuration file using the end point of the previous one (an example in submissionScript.sh for SLURM scheduling is provided).

Worked Example

Alongside this tutorial, a .zip file is available that contains all the files needed to run a simulation of insulin, the smallest protein (and therefore also the easiest to simulate locally!). The aim of this section is for you to be able to create all the files contained in this folder without having to open it, but it should be a useful resource to compare and troubleshoot with no matter how it's approached. It is worth noting here that the simulations undertaken here are designed to be run on your local machine without too much waiting – the times taken are woefully small for a true MD simulation of biomolecules, which typically expect at least 1ns of equilibration time and 100ns of dynamics. Before we can get into the first step, however, we need to make sure we have NAMD installed and the CHARMM36 parameters available. The installation of NAMD is fairly simple – download the folder containing the executable, and make sure your path has access to it (I have a 'namd2' and 'psfgen' file in my /usr/local/bin directory that points toward the NAMD/psfgen executable in my Applications folder); once you've done this for NAMD, repeat for VMD. Installation of the CHARMM36 parameters is even easier – they simply need to be downloaded; when you make your psfgen and configuration files, you provide a path to these. In the bundled example files, the parameters are provided in the same folder. With these programs installed, you now need to obtain the starting geometry – we'll be using the '3I3Z' crystal structure from the Protein Data Bank, which is accessible by the provided hyperlink, or by searching 3I3Z into the PDB website. Download the file in .pdb format, and then we're ready to get started!



The first step in our process is to run the psfgen command to create a file with the bonding arrangement of our protein. Psfgen is a useful program, but it is also quite particular about what is required despite barely explaining these requirements. The easiest way to demonstrate this is with an example – and so the script used on the insulin molecule follows:

```
package require psfgen                                          [1]

topology top_all36_prot.rtf                                     [2]
topology toppar_water_ions_namd.str

pdbalias atom ILE CD1 CD                                        [3]
pdbalias residue HIS HSE

segment A {                                                     [4]
pdb 3i3z_prepared_a.pdb
}

segment B {
pdb 3i3z_prepared_b.pdb
}

patch DISU A:6 A:11                                             [5]
patch DISU A:7 B:7
patch DISU A:20 B:19

coordpdb 3i3z_prepared.pdb                                      [6]

guesscoord                                                      [7]

writepsf 3i3z_psfgen.psf                                        [8]
writepdb 3i3z_psfgen.pdb
```

The start of the file (section [1]) simply ensures that psfgen is being used, as these scripts can be run through VMD as well. Section [2] provides the path to the topology files required (the protein and water topologies respectively). Section [3] is a little more complicated – changes to the .pdb file are required, as there are some differences between the .pdb files used by crystallographers and those used in MD. The histidine amino acid can be in one of two protonation states – both are called HIS in crystal structures, as the hydrogen atoms are not commonly (yet) resolved experimentally. These two lines change the names of atoms or residues in .pdb files to those used by CHARMM – and will likely be required in all MD simulations of proteins you run using CHARMM force fields. Section [4] details the various segments of the molecule - as the molecule is processed on a chain-by-chain basis, each chain in the .pdb file will require a different segment in the psfgen file. Inside of the segment you must provide a .pdb file for that chain (a small VMD script is supplied, splitABScript.tcl, that will generate these automatically while removing crystallographic water), and specific changes can be given – changing specific atom names, changing the capping groups of the chain, and so on. The script to split the insulin protein can be called using the following command:

```
vmd -dispdev text -e splitABScript.tcl
```

Section [5] details any changes you want to apply to the system that are not specific to a single chain – the examples listed are forming disulfide bonds between non-sequential residues in the chain (A6-A11, A7-B7, and A20-B19). If you're interested in the changes available, a list of all available changes here is found by searching for PRES in the topology files being loaded. Moving towards the end of the file, section [6] details the .pdb file containing the atoms of the entire system, and section [7] instructs psfgen to guess the location of any atoms that aren't present in that .pdb (for example, in the .pdb file being used, hydrogen atoms are omitted) – this is done based on the residue name, which is why section [3] picks a histidine residue name – this is essentially specifying protonation. Finally, section [8] details the output – writing the new .psf and .pdb files. Once you're happy with the way the script works, you can run it using the psfgen executable:

```
psfgen psfgen.pgn -> psfgen.log
```

If the log isn't showing errors, I'd advise loading up the resulting structure in VMD and inspecting it to make sure it looks as expected:

```
vmd 3i3z_psfgen.psf 3i3z_psfgen.pdb
```

If there aren't any obvious errors (and psfgen errors are typically obvious – bonds far too long, or more commonly a cluster of atoms with 0,0,0 x,y,z coordinates), you're now safe to move on to the next step. If these errors do occur, carefully check the generated log file from the above step – the error should be stated here. These errors are typically due to an error in the script itself – neglecting the segments, applying patches with the wrong syntax, or incorrect paths to topologies are all common mistakes. If the generated structure is incorrect but there is no error in the psfgen log file, it is likely caused by the initial structure

– check that this structure is sensible, and does not have excessively close contacts, non-physical bond distances, or the like.
VMD is used to solvate and then ionize the protein – this can be done inside of VMD's GUI (through the Extensions -> Modelling -> Add Solvation Box/Ions menu) or via a script. The use of the modelling menu opens a small window – most options are fairly self-explanatory. The only changes needing to be made in the solvation menu is the addition of 12Å of box-padding – the minimum distance from all points in the protein to the edge of the solvation box. When ionizing the solvation box, you simply need to choose the option to neutralize and add salt to a concentration of 0.2 M (mimicking experimental conditions, in this case). Doing this process through a script is simple (and I would recommend it – simply because it's consistent and you can easily check what you did):

```
package require solvate                                                    [1]
package require autoionize

set inputName 3i3z                                                         [2]
set psfgenTerm _psfgen
set psfgenName $inputName$psfgenTerm
set solvateTerm _solvated
set solvateName $inputName$solvateTerm
set ionizedTerm _ionized
set ionizedName $inputName$ionizedTerm

solvate $psfgenName.psf $psfgenName.pdb -t 12 -o $solvateName              [3]

# Need to neutralize as PWE electrostatics require a neutral system, so
#properly done PBC needs a neutral system
autoionize -psf $solvateName.psf -pdb $solvateName.pdb -sc 0.2 -cation SOD
-anion CLA -o $ionizedName                                                 [4]

set backbone [atomselect top "backbone"]                                   [5]
set all [atomselect top all]
$all set beta 0
$backbone set beta 1
$all writepdb 3i3z_ionized.pdb

exit
```

Both sections [1] and [2] are simple – section [1] states the packages used for VMD, and section [2] defines variables to allow output in the correct format. Section [3] solvates the previously created structure from psfgen; the only unexpected argument is -t 12, which creates a buffer of at least 12 Angstroms from the solute to the edge of the solvent box. Section [4] controls the ionization process, with the -sc argument controlling the concentration of ions, and SOD and CLA being the CHARMM residue names for sodium and chlorine ions. Finally, section [5] makes a small modification to the final .pdb file – the beta value of all atoms is set to 0, then the beta value of the backbone atoms of the protein chain are set to 1, and the changes are written to the .pdb file. This is done to facilitate constraining the protein in the next step. There's another section to the script supplied, but it's not overly relevant – a small function is defined that outputs the size of the final box for

the configuration file used in the next step (see Extra Information). Once you're happy with how this works, the script can be run the same way the previous tcl script was:

```
vmd -dispdev text -e solvateIonizeScript.tcl
```

Now that we've finished solvating and ionizing the system, we've finally got our system ready to go! To prevent a large number of files building up in one folder, we've got a different folder for the different equilibrations that are oncoming, as well as for the final production run, so we need to copy the final system created here over:

```
cp *_ionized* ../equilibration/
cp *_ionized* ../productionRun/
```

We can't just launch straight into a production run intended to generate results in MD – we've got to minimize the structure, and equilibrate the system. The exact process one uses to do this differs depending on what you're doing in your system, but the general trend stays the same across most biomolecules – allow the solvent to equilibrate, allow the box to fluctuate to its preferred size, and allow the solute to equilibrate. Running a simulation based on a configuration file is simple:

```
namd2 +p(number of cores to be used) /path to configuration file/ > /path
to desired output log file/

e.g. namd2 +p4 prodRun.conf > prodRun.log
```

For this example, we'll look through the configuration files in order – a period in the canonical (NVT) ensemble with the protein frozen will allow the equilibration of the solvent, a period in the NPT (isothermal-isobaric) ensemble will allow the box size to fluctuate, and a period in the canonical ensemble with protein restraints being reduced will allow the solute to equilibrate. While we're discussing ensembles, it's worth mentioning that for the final production run, there's no strong agreement on what ensemble should be used – some groups use an NPT ensemble to most closely resemble experimental conditions, whereas others use the NVT ensemble as it is slightly cheaper and more reliable; this is of course a very broad summary of a rather long dispute. The files used in the equilibration are called solventNVT.conf, solventNPT.conf, and soluteNVT.conf for the corresponding stages outlined above. We'll go through the first of these files in detail, then explain any new additions for the remaining files. The first area of interest is the relatively mundane opening section:

```
# NAMD configuration file for an insulin CHARMM36 NVT ensemble run at the
beginning of a simulation                                              [1]

# Declaring of global variables                                        [2]
set temperature 298

# Initial structures from previous steps - ionized and solvated - can have
restart info here                                                      [3]
structure 3i3z_ionized.psf
```

```
coordinates 3i3z_ionized.pdb

# Output                                                                    [4]
binaryOutput no ;# Don't want binary format, not pdb, for output
binaryRestart no ;
outputName 3i3z_solventEquilibrated
DCDfile 3i3z_solventEquilibrated.dcd
restartfreq 5000
dcdfreq 1000
xstFreq 5000
outputEnergies 100
outputTiming 1000 ;# Typically want x10 outputEnergies

# Forcefield (paratypeCharmm specifies Charmm-style parameters, as opposed
to others that NAMD can use)                                                [5]
paraTypeCharmm on
parameters par_all36_prot.prm
parameters toppar_water_ions_namd.str
```

It can be seen here that much of this initial part of the file is dedicated towards simple setup of the simulation. Section [1] is a simple title for the file (making it easier to open and see what you did without looking through the whole thing), Section [2] declares the only global variable required for this run (so you don't need to have temperature set in several locations), and Section [3] simply provides the path to the .psf and .pdb files generated earlier. Section [4] details the output of the simulation – turning off binaryOutput and binaryRestart allows human-readable output files, outputName and DCDfile are the output names for the restart output and trajectories respectively; all remaining options here are the frequencies (in number of steps) of the output of restarting files, trajectories, box information, energies, and benchmarking, respectively. The next section of the configuration file deals with the more theory-focused options:

```
# Approximations - the higher these cutoff distances, the more accurate -
these are cutoffs that are accurate for protein/biomolecular systems   [6]
switching on
switchdist 10 # 2 Angstrom below the cutoff distance
cutoff 12 # Should be at least a few Angstrom beyond longest reasonable
long-range interactions in system
pairlistdist 14 # pairlistdist - cutoff should be at least the distance
that can be moved in one cycle, as defined in stepspercycle
stepspercycle 20
exclude scaled1-4
1-4scaling 1.0

# Electrostatics                                                            [7]
PME yes
PMEGridSpacing 1 ;# A single control for the size of the PME grid –
smaller is more accurate, but more costly and very small (below 1) values
may cause instabilities
```

```
# Integrator                                                          [8]
timestep 2.0
rigidBonds all
nonbondedFreq 1
fullElectFrequency 2

# Constant Temperature Control                                        [9]
langevin on
langevinDamping 1.
langevinTemp $temperature
langevinHydrogen no
```

In section [6], the approximations that are used are listed. Switching is a gradual reduction of non-bonded forces at the listed distance, leading to a complete cut-off at the cutoff distance. The pairlistdist, stepspercycle, and 1-4scaling properties are thoroughly defined in the NAMD documentation and are explained in more detail there – in short, the first two define a region just outside the cut-off where atoms are monitored for entrance into the cut-off region, and the final parameter scales the non-bonded interactions between atoms separated by 3 bonds. Section [7] is simple – it enables Particle Mesh Ewald summation for the long-distance electrostatic interactions, and then allows NAMD to choose the grid size. Section [8] deals with the integrator – the timestep is set to 2fs, rigidBonds are set to all hydrogen-heavy atom bonds, and the frequency of non-bonded calculations are set here. Finally, Section [9] controls the thermostat the regulates temperature – it is set to a Langevin thermostat, controlled to the set temperature, and doesn't affect hydrogen atoms (used in conjunction with rigidBonds in section [8], this means hydrogen atoms' dynamics are controlled by the heavy atom to which they are bonded). We then get to the down to earth options:

```
# Periodic Boundary Conditions - vector1/2/3 is the vector to the next
image, cellorigin is the center of the cell; see the get_cell script in
the NAMD tutorial or set of scripts to generate these                [10]
cellBasisVector1 47.691999435424805 0 0
cellBasisVector2 0 47.525999546051025 0
cellBasisVector3 0 0 55.8179988861084
cellOrigin -9.467723846435547 -17.910892486572266 -0.35832399129867554

wrapWater on ;# Wrap water molecules
wrapAll on ;# Wrap all non-water molecules
wrapNearest off ;# This is used for non-rectangular or curved cells

# Constrained Atoms - a harmonic restraint, typically applied to the
solute of the system to allow solvent equilibration                  [11]
constraints on
consexp 2 ;# The exponent used in the harmonic constraint
consref  3i3z_ionized.pdb ;#
conskfile 3i3z_ionized.pdb ;# The file where the force constant for the
atom's restraining potential is found - can be same as above
conskcol B ;# Use the beta column of the above file
constraintScaling 250 ;# Constraints for the minimization
```

The first section here, Section [10], details the periodic boundary conditions. The first part contains the three vectors and the central point of these vectors; together, these create the box. These vectors are created automatically by the get_cell function of the solvation script – they are found in the file boxSize.rtf, and the contents of this file can be copied directly to this area without issue. The second half controls what atoms can move from one side of the solvation box to the other – all molecules, water and not-water are included. Section [11] contains the commands related to constraining the atoms – initially it enables constraints and provides the files in which the constraints should be applied (consref) and the list of atoms that should be constrained (conskfile). The conskcol command details how the list of atoms is provided – in this case, it's the beta column of a .pdb file, where any non-0 number present in this column will enable constraints. The constraintScaling parameter defines the strength of these constraints in kcal/mol. With this completed, we just have the final section left:

```
# Protocol                                                                    [12]
temperature $temperature
minimize 50000
reinitvels $temperature ;# Needed after minimization
run 50000
```

This section is very simple – it generates velocities for atoms at the set temperature, minimizes for a set number of steps, reinitiates the velocities, and runs for a set number of steps. With this section completed, we've run through the whole configuration file, and the file can be run:

```
namd2 +p4 solventNVT.conf > solventNVT.log
```

Before we move on from configuration files, however, we need to discuss the changes that occur in the next step – NPT equilibration; they are fairly minimal, with a Constant Pressure Control section being the sole addition:

```
# Constant Pressure Control (allows the volume to vary - i.e. makes the
simulation NPT not NVT)
useGroupPressure yes ;# Without this, can't have the rigid hydrogen bonds
needed for a 2fs timestep
useFlexibleCell no ;# Not needed for a waterbox, if using a membrane
protein it is needed
useConstantArea no ;# Not needed for a waterbox - for a membrane protein,
use in a case-by-case basis
langevinPiston on
langevinPistonTarget 1.01325 ;# 1 atm pressure in bar
langevinPistonPeriod 100. ;# Oscilation period around 100 fs is standard
langevinPistonDecay 50. ;# Oscilation decay around 50fs - want it smaller
than the Period to ensure harmonic oscillations are overdamped
langevinPistonTemp $temperature ;# Coupled to the heat bath
```

This section allows the variation of the volume of the box through the addition of a barostat to control pressure. The flexible cell command allows each length of the box to vary independently of the others, which is not required in a standard water-box system – this is

most commonly used for simulations of membrane systems. The langevinPiston is the method through which this occurs; it has a set target (1 atm normally), which oscillates over a set period of time. This period should be kept fairly long, as pressure will naturally vary quite dramatically over time in MD simulations. This equilibration can be run in a very similar way to previously:

```
namd2 +p4 solventNPT.conf > solventNPT.log
```

Outside of the addition of the pressure control section, the other change in the remaining configuration files occurs in the soluteNVT.conf file, and details the reduction of constraints on the protein - this requires a little tcl scripting in the protocol section to do so in a safe way. The sudden release of strong constraints – 250 kcal/mol – on the protein can cause instability, and so it is instead reduced in successive bouts of smaller restraints:

```
# Protocol
run 5000
for {set i 0} {$i <= 3} {incr i 1} {
        set ck [expr {250.0 *(1.0/(10.0**$i))}]
        constraintScaling $ck
run 5000
} ;# This one is a bit of a little trick - run for 5000 steps, then
decrease the energy of the constraints by 10x, and repeat till at 0.25
kcal/mol, then run without any constraints for a while to allow
equilibration of the solute
run 50000
```

As can be seen above, the protocol section from the soluteNVT.conf file is a little different from the previous two – a small for loop is created in the TCL scripting language in which the constraintScaling parameter is exponentially reduced as the loop is continued, and short sets of steps are run. Once this loop is completed, the simulation runs for a nanosecond of NVT equilibration without any restraints. This equilibration can then be run:

```
namd2 +p4 soluteNVT.conf > soluteNVT.log
```

With this completed, the protein is fully equilibrated and ready for a production run – the actual simulation. You'll need to copy the output of the final equilibration over to the production run folder:

```
cp *soluteEquilibrated* ../../productionRun/
```

The production run is very simple – it uses a configuration file that's almost identical to the initial solventNVT file with the constraints removed, and a much longer run time. Have a read through the prodRun.conf file – if you're not sure what anything does, it should be explained above (or in the NAMD documentation, if you'd rather get used to it now). Then, when you're ready, run the production run and get ready to analyze it:

```
namd2 +p4 prodRun.conf > prodRun.log
```

Analysis

With the above completed, you now have a final trajectory available to you. The difficult step is how you analyze it! There are many possible forms of analysis in MD, and what is useful or not is dependent on the system you're analyzing. There are some easy ones that are fairly universally applicable: calculation of the density of the system, RMSD of the system, energies in the system, and the like. Others are a little more complicated to do – hydrogen bond analysis, secondary structure analysis, or the value of bonds/angles/dihedrals over the course of a simulation. Shown below is a table of some analysis methods available, and what programs are able to undertake this analysis:

Analysis Method	Program Used	Common Motivation
RMSD	VMD/CPPTRAJ/MDTRAJ	Ensure the system has equilibrated – the RMSD isn't still increasing
Energies (also covers pressure/energy contributors/volume and temperature)	NAMD/VMD	Validate the system – ensure no strange energies are occurring, and everything is staying how it should
Density	VMD/MDANALYSIS	Ensure the parameters are correct via a comparison to experimental density
Hydrogen Bond Analysis	CPPTRAJ/MDTRAJ	Determine the degree of interaction between residues
Secondary Structure Analysis	VMD/CPPTRAJ/MDTRAJ/ MDANALYSIS	Validation through comparison to experiment, or determination of properties
Native Contacts/Closest Atoms	CPPTRAJ/MDTRAJ/ MDANALYSIS	Determination of the degree of interaction between two residues
Distances/Angles for sets of atoms	VMD/CPPTRAJ/MDTRAJ	Tracking over the course of a simulation – make into a histogram or similar
Radius of Gyration	MDTRAJ	Calculate the degree of unravelling/disorder of the solute
Solvent shell extraction	CPPTRAJ/MDTRAJ	Analyze the degree of solvation and geometries of solvation shells
Generation of Ramachandran Plots	CPPTRAJ/MDANALYSIS	Validation by comparison to known values, or determination of these values for new residues
Cluster Analysis	CPPTRAJ/MD ANALYSIS	Ascertaining the most common states accessed

2D RMSD	CPPTRAJ	Comparing the RMSD of two separate segments
Principle Component Analysis (PCA)	CPPTRAJ/MDANALYSIS	Complicated, but can identify the most 'important' movements in a simulation
Elastic Network Analysis	MDANALYSIS	Identification of the conformational states of a biomolecule
Water Dynamics Analysis	MDANALYSIS	Identification of the interactions of water with solute, and behavior of water as a solvent
Calculation of NMR Observables	MDTRAJ	Comparison to experiment primarily
Thermodynamic Values (dipoles, dielectrics, etc.)	MDTRAJ/CPPTRAJ	Either comparison to experiments or information on environment of area

As you can see from the table above, there is a fairly large range of programs that can be used in the analysis of MD trajectories – most of these try to be fairly MD package agnostic, and can be used regardless of whether you used AMBER, GROMACS, NAMD, and so on. From the programs listed above, MDANALYSIS and MDTRAJ are both python libraries, CPPTRAJ[7] is part of the AmberTools package and is a dedicated program, and VMD[8] is a visualization program. Other analysis tools also exist – TRAVIS is commonly used for both analysis and visualization in the group for LAMMPS simulations, for example, but the above tools should be sufficient for almost any NAMD analysis of interest. Visualization of the results of these results can be carried out in a variety of ways – languages such as R, or the matplotlib package for python work well for presentation of graphs and the like, and VMD can present a visual representation effectively. An example of this analysis is provided along with the example above – the cppTrajAnalysisScript.in file. Initially, the script has to deal with some technical aspects to get the finished script loaded:

```
# Set variables for input/output, and load in the raw input            [1]
set inputParm=3i3z_ionized.psf
set inputTraj=3i3z_prodRun.dcd
set outputParm=3i3z_stripped.psf
set outputTraj=3i3z_prodRun_stripped.dcd
parm $inputParm
trajin $inputTraj

# Strip all atoms that have the resname TIP3 (the TIP3 water model used),
SOD (Sodium), or CLA (Chlorine), then output the modified topology and
trajectory                                                             [2]
strip :TIP3,SOD,CLA outprefix strip nobox
trajout $outputTraj
run
```

```
# Once the above has run, the topology is misprinted - as an Amber file,
despite the .psf extension, so reload it write it out explicitly as a
CHARMM-readable X-PLOR PSF                                              [3]
# I.e. the issue is the outprefix option of the strip command outputs as
Amber no matter what, unlike the parmwrite out command
parm strip.$inputParm
parmwrite out $outputParm 1

# Reset back to initial state, as the solvent + ions are stripped and
correct files are present, then load in the to-be used files            [4]
clear trajin
clear parm
clear trajout
parm $outputParm
trajin $outputTraj
```

Initially a set of variables are declared containing the names of the input trajectory and 'parameter' – the .psf file. Once this is completed, these files are loaded in. The first step (section [1]) is to strip the solvent and ions from the trajectory – these make up the majority of the atoms in the system, and unless you are running analysis that includes solvent atoms explicitly, they've served their purpose by solvating the molecule of interest and can be safely removed. Once this is complete, you run the above commands – i.e. actually strip it, shown in section [2]. Due to a bug in CPPTRAJ, this is output as an amber-style trajectory, not a .dcd file – so you load it in one more time and explicitly write it as a NAMD .dcd file, and reset to the default state – done by section [3]. With this completed, you load in the now-stripped trajectory in section [4], and you are ready to start analysis.

```
#Begin actual analysis here
# First is distance - calculating the distance between Cys6/11 and Cys7/28
of just the sulfur atoms to the non-backbone parts of the HSD residues.
This is done by nativecontacts to facilitate doing the minimum distance
between any atoms involved, not between centre of masses.               [5]
nativecontacts name Cys6His26 :6,11@SG,CG :26&!@CA,C,O,N,HA,HN mindist out
Cys6His26Raw.dat
# Hbond analysis is next - for comparison to the Insulin in Motion paper,
so just focusing on A8's NH h-bonds with A3-5's CO group                [6]
hbond :3,4,5,8&@C,O,N,HN out hbond.dat avgout avghbond.dat
# Secondary Structure analysis comes next, focused on A3-A8             [7]
secstruct A3 out A3A8DSSP.dat sumout A3A8DSSP.out :3-8 nameh HN
# Calculate distance between the A6-A11 alpha carbons                   [8]
distance A6A11Distance :6@CA :11@CA
run

# Further analysis here relies on the previous steps having been
completed, thus the run line above
# Convert the minimum distance files to histograms                      [9]
hist Cys6His26[mindist] min 0 max 15 step 0.1 norm out
Cys6His26Histogram.agr
# Convert the distances between the alpha carbons of the disulfide linkage
to averages
avg A6A11Distance A7B7Distance out alphaCarbonDistances.dat
```

```
# Cluster analysis - the most complex of the lot, so will space out
commands with \ for improved readability                              [10]
cluster InsulinCluster \
        dbscan minpoints 6 epsilon 1.1 sievetoframe \
        rms :1-51&@N,CA,C,O \
        sieve 10 random \
        out cnumvtime.dat \
        summary summary.dat \
        info info.dat \
        cpopvtime cpopvtime.agr normframe \
        repout rep repfmt pdb \
        singlerepout singlerep.pdb singlerepfmt pdb
run
# Can't easily comment each line, but using the DBSCAN clustering method -
density based, can discard points, with minpoints and epsilon determined
using a previous kdist
# This using a distance metric of RMS on all non-hydrogen atoms (in the
stripped topology), drawing out 1/10 (sieve) to start with then building
from there
# Outputting the number of clusters over time, a summary, a detailed set
of information, the population vs time (normalized by frame), a
representative PDB of each cluster and all of those PDBs in a single PDB -
singlerepout.

# Run the RMSD calculation last, as it moves the co-ordinates of the atoms
involved - running on all residues (1-51) and ignoring all hydrogens. [11]
rms RMSD :1-51&@N,CA,C,O first out rmsd1.agr mass
```

CPPTRAJ allows the queuing of commands, essentially – all commands can be input, and only when the run keyword is encountered does the analysis begin. Most of the steps here have been reduced to a single example, and so are relatively short – in a more formal analysis, it is likely you'll have several distances, closest interactions, areas of secondary structure, and so on that are of interest. Section [5] calculates the minimum distance between the two listed groups – the sulphur/carbon atom in the A6-A11 linkage, and the imidazolium ring in HisB5, and outputs this as a human-readable .dat file – for each frame, the shortest distance is output. Section [6] details the hydrogen-bonding analysis of the backbone of a short stretch of protein and a specific residue, putting the frame-by-frame results in one file and the average over the simulation in another. Section [7] follows in a similar area, performing an analysis of the secondary structure of the protein in this region, once more outputting the average and frame-by-frame results to separate files. Section [8] is exceptionally simple – it outputs the distance between the alpha carbons of the A6 and A11 residues over the course of the simulation, but it doesn't output this to a file. Section [9] runs after the above have completed, and has 2 similar lines – the first converts the raw data generated from section [5] into a histogram, and the second takes the raw data from section [8] and averages it across the simulation, which is then output.

Section [10] is the most complicated of these, and is responsible for clustering the simulation, with a variety of options. The comments below it should explain the options in more detail, but the rough version of it is that it clusters the simulation based on the RMSD of the backbone atoms of the protein using the DBSCAN method – a minimum number of points must be close to each other for this to detect a cluster, and points may be excluded if they fit into no cluster. A variety of output statistics are made, the most relevant being singlerep – a .pdb file that is representative of each cluster – and summary, which contains information on the proportion of frames in each cluster, and the standard deviation within that cluster. Finally, section [11] is very simple – it outputs the result of RMSD over time on the non-hydrogen backbone atoms of the protein. With that done, all the analysis being undertaken is completed – if you're interested in this type of analysis, browsing these created files is likely an informative use of time.

The analysis above is focused entirely on CPPTRAJ, but the documentation for the alternative MD analysis programs are linked above – a perusal of their documentation demonstrates that the techniques applied are similar. Deciding on what analysis to be done is often the trickiest part of MD – you have a huge simulation over a long time period, but you need to extract useful information from it. There's not much one can do to 'teach' this – you just need to think about what you want out of the simulation, and to look at the tools you have available. I'd also recommend reading a variety of MD papers in your area of research – the techniques used tend to be quite similar across different experiments, so seeing what is used in the specific area you're investigating is of considerable use.

Before we end, I just have a few strange and specific tips that didn't fit in earlier that I'd like to pass on! The program catdcd comes with a download of NAMD and is a useful concatenating tool for DCD files – it's very useful if you need to restart a simulation several times, as it allows you to combine them together for analysis. Secondly, VMD is an excellent tool for visualization and presentation, and there are several small changes I've found to preparing figures in VMD. Initially, I'd get rid of the axes (Display -> Axes -> Off) and change

the background color to white (Graphics -> Colors -> Display -> Background -> 8 white). The NewCartoon representation (Graphics -> Representations -> Drawing Method -> NewCartoon) works well for proteins and biomolecules, showing their overall structure without overwhelming the viewer with visual information – and in the same screen, changing the material to Glass1/2/3 or Transparent can work well to de-emphasize the overall structure in favor of the area of interest you're showing. A representation of your area of interest works well in the licorice Drawing Method, or CPK if you're going for more similarities with a typical Avogadro-style representation. Occasionally some of the renderers have trouble with overlapping representations, in which case increasing the bond radius of the representation you want shown will differentiate them sufficiently for the renderer. Once you're happy with how the figure looks in the display window, it's time to render: File -> Render. My recommendation is to render using Tachyon (internal) rendering – it tends to give the crispest looking picture (output in the .tga format), while rendering quickly for large scenes. Remember that the level of zoom you're using in the Display window will be recreated in the render, so frame your image as you want it in the final shot!

Extra Information

Forcefield Advantages

A more thorough discussion of the differences between various types of force fields available in NAMD is potentially useful. The CHARMM forcefields are the most commonly used alongside NAMD, and besides the most modern version, the only commonly used forcefield is CHARMM22* - a modification of CHARMM22 to better fit backbone dihedrals; it generally performs well on unfolded proteins and in protein folding simulations. CHARMM36m is the newest version and has been shown to work better for non-protein biomolecules and folded proteins – in addition, as the more modern version, more tools are available for it. In the set of Amber force fields, there's more variation: the Ff14SB force field is a classic Amber force field – building off of a set of force fields since their 1999 release, the details of the forcefield are well known and a variety of older tools are still compatible. Outside of these benefits, it's been overtaken in many regards by FF15FB – a slightly more recent forcefield that's empirically derived and doesn't rely on older parameters as much, and excels in modelling bonded interactions. Their Ff15ipq forcefield is a different one altogether – it uses a technique they've created to automate the majority of the *ab initio* calculations required for the generation of the forcefield, and so doesn't rely on older values whatsoever. It typically outperforms the Ff15FB forcefield at non-bonded interactions but isn't as impressive at bonded interactions. GROMOS force fields are an easy discussion – there aren't many of them, and not simultaneously; the GROMOS 54a8 forcefield is the most recent and is recommended. More detailed information is available in the references, with each forcefield having a paper discussing the parameterization and advantages in detail.

Solvation Boxes and Periodic Boundary Conditions

Solvation boxes default to a square or rectangular shape for a good reason - it is possible to solvate in other shapes, with spherical solvation even minimizing the solvent atoms:solute atoms ratio (and so maximizing cost efficiency of the calculation), but these are rarely used. The evaluation of long-range ionic interactions is difficult computationally, as they reduce in intensity across distance considerably more slowly than the Lennard-Jones model of non-coulombic interactions. Almost all modern MD packages use either a Particle Mesh Ewald (PME) method to solve this or a very similar technique – after a set distance, a grid is

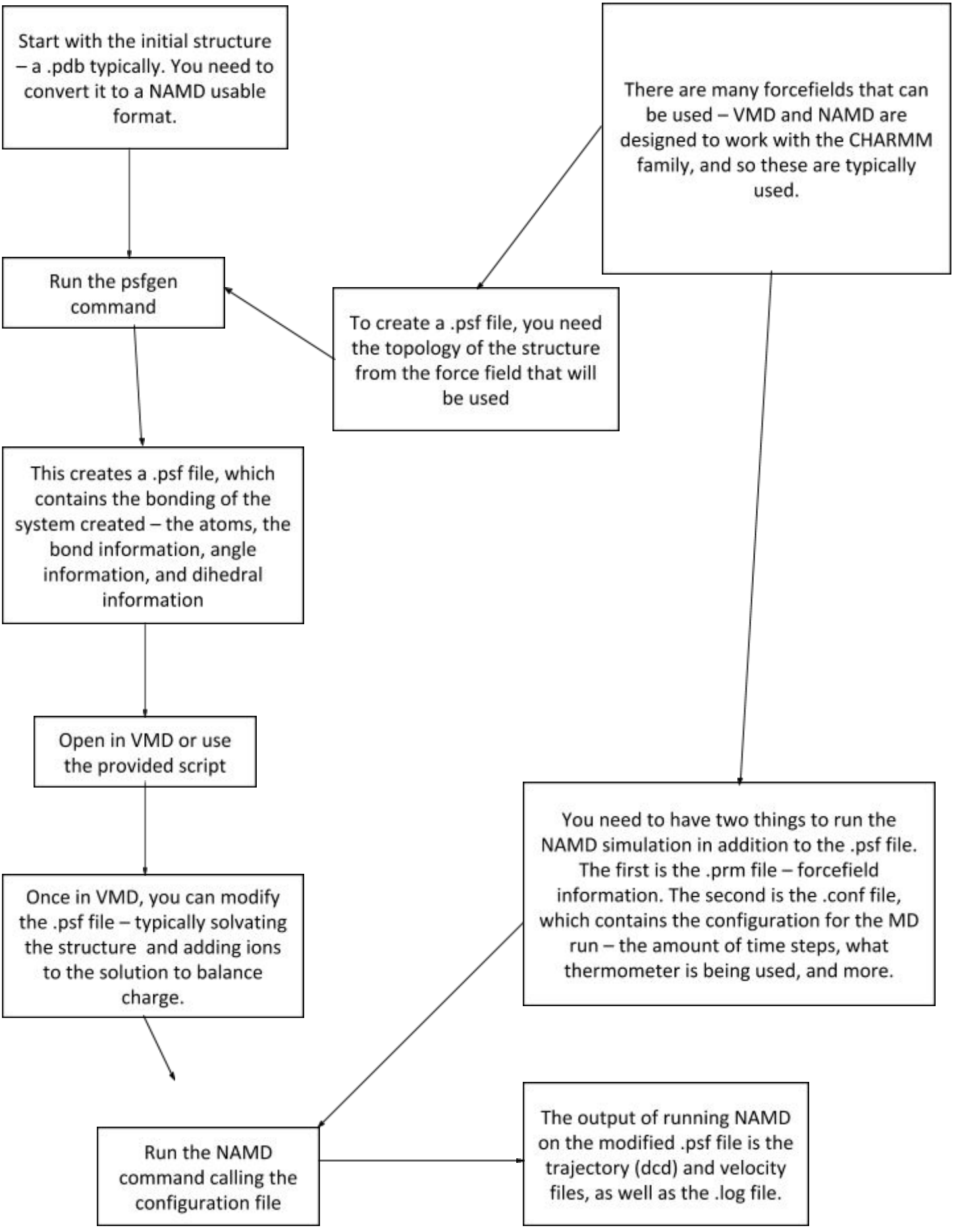
created (typically in 1 $Å^2$ grid sizes for NAMD) and the sum of all charges in this grid is used for the interaction. This is computationally considerably cheaper, but this process continues until no new charges are being found – this requires a net neutral system, and periodic boundary conditions (PBC) to be in effect. Periodic boundary conditions involve atoms at the edge of the solvent box being affected by the forces of atoms on the exact opposite side of the box – if an atom travels out of the box, it is instead moved to the opposite side of the box. This effectively eliminates boundary effects in which the solute experiences a different environment at the edge of the box, but this necessitates a rectangular solvation box.

Configuration File Options

Command	**Requirement**	**Effect**
structure	A path to a .psf file	Informs the simulation of the connectivity of atoms present
coordinates	A path to a .pdf file	Informs the simulation of the xyz location of atoms present
restartfreq	An integer	How often (in number of steps) the full set of restart files are output
parameters	A path to a forcefield parameter file	Informs the simulation of the parameters for atoms in the simulation; multiple of these are almost always required
cutoff	An integer	The point at which non-coulombic long-range interactions are cut off
PME	A Boolean yes/no	Enables Particle Mesh Ewald summation for coulombic interactions (see extra information)
timestep	Floating point	The amount of time (in femtoseconds) the simulation jumps in-between each step; longer is more efficient, but may miss more information
Langevin	A Boolean on/off	Enables Langevin dynamics to control the temperature of the simulation
cellBasisVector1/2/3	A vector of 3 floating points	There are three of these – together with the cellOrigin, these vectors define the 3d space that is the simulation box; the get_cell script will define these for you
wrapAll	A Boolean on/off	Enables all molecules to ‘wrap’ – move from one end of a simulation box to the other when under periodic boundary conditions
langevinPiston	A Boolean on/off	Enables a Langevin piston that controls the pressure of the simulation in addition to the temperature; this allows the NPT ensemble to be used
constrains	A Boolean on/off	Allows the constraining of atoms with a harmonic potential; useful to allow things such as the solvent equilibrating without interference from the solute, or vice versa

temperature	A floating point	Sets the temperature of the simulation to this value (in K)
minimize	An integer	Starts a minimization; this is a minimization by steepest descent by default and runs for the given number of steps. Once a simulation has started, many options cannot be changed – so put this at the end of your configuration file.
run	An integer	Starts a simulation that runs for the given number of steps; again, this locks in many options and so should be done at the end of a configuration file.

NAMD Flowchart



The get_cell script

```
proc get_cell {{molid top}} {
 set file [open boxSize.rtf w]
 set all [atomselect $molid all]
 set minmax [measure minmax $all]
 set vec [vecsub [lindex $minmax 1] [lindex $minmax 0]]
 puts $file "cellBasisVector1 [lindex $vec 0] 0 0"
 puts $file "cellBasisVector2 0 [lindex $vec 1] 0"
 puts $file "cellBasisVector3 0 0 [lindex $vec 2]"
 set center [measure center $all]
 puts $file "cellOrigin $center"
 $all delete
 close $file
}
```

References

(1) Huang, J.; Mackerell, A. D. CHARMM36 All-Atom Additive Protein Force Field: Validation Based on Comparison to NMR Data. *J. Comput. Chem.* **2013**, *34* (25), 2135–2145. https://doi.org/10.1002/jcc.23354.

(2) Buck, M.; Bouguet-Bonnet, S.; Pastor, R. W.; MacKerell, A. D. Importance of the CMAP Correction to the CHARMM22 Protein Force Field: Dynamics of Hen Lysozyme. *Biophys. J.* **2006**, *90* (4), L36–L38. https://doi.org/10.1529/biophysj.105.078154.

(3) Reif, M. M.; Winger, M.; Oostenbrink, C. Testing of the GROMOS Force-Field Parameter Set 54A8: Structural Properties of Electrolyte Solutions, Lipid Bilayers, and Proteins. *J. Chem. Theory Comput.* **2013**, *9* (2), 1247–1264. https://doi.org/10.1021/ct300874c.

(4) Maier, J. A.; Martinez, C.; Kasavajhala, K.; Wickstrom, L.; Hauser, K. E.; Simmerling, C. Ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from Ff99SB. *J. Chem. Theory Comput.* **2015**, *11* (8), 3696–3713. https://doi.org/10.1021/acs.jctc.5b00255.

(5) Debiec, K. T.; Cerutti, D. S.; Baker, L. R.; Gronenborn, A. M.; Case, D. A.; Chong, L. T. Further along the Road Less Traveled: AMBER Ff15ipq, an Original Protein Force Field Built on a Self-Consistent Physical Model. *J. Chem. Theory Comput.* **2016**, *12* (8), 3926–3947. https://doi.org/10.1021/acs.jctc.6b00567.

(6) Wang, L. P.; McKiernan, K. A.; Gomes, J.; Beauchamp, K. A.; Head-Gordon, T.; Rice, J. E.; Swope, W. C.; Martínez, T. J.; Pande, V. S. Building a More Predictive Protein Force Field: A Systematic and Reproducible Route to AMBER-FB15. *J. Phys. Chem. B* **2017**, *121* (16), 4023–4039. https://doi.org/10.1021/acs.jpcb.7b02320.

(7) Roe, D. R.; Cheatham, T. E. PTRAJ and CPPTRAJ: Software for Processing and Analysis of Molecular Dynamics Trajectory Data. *J. Chem. Theory Comput.* **2013**, *9* (7), 3084–3095. https://doi.org/10.1021/ct400341p.

(8) Humphrey, W.; Dalke, A.; Schulten, K. VMD: Visual Molecular Dynamics. *J. Mol. Graph.* **1996**, *14* (1), 33–38. https://doi.org/10.1016/0263-7855(96)00018-5.