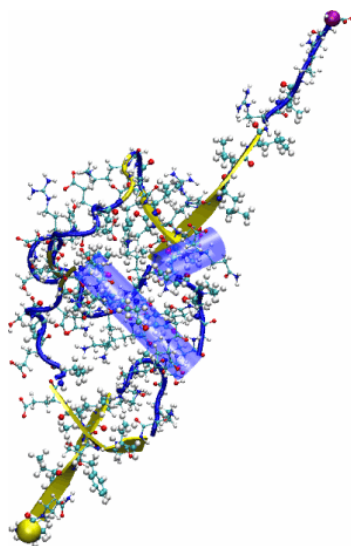


University of Illinois at Urbana-Champaign  
Beckman Institute for Advanced Science and Technology  
Theoretical and Computational Biophysics Group

# NAMD TUTORIAL

---



NAMD Developer: James Phillips

Timothy Isgro

James Phillips

Marcos Sotomayor

Elizabeth Villa

May 2003

A web version, in color, is available at  
<http://www.ks.uiuc.edu/Training/SumSchool03/Tutorials/namd>

# Contents

<b>1</b>	<b>Basics of NAMD</b>	<b>5</b>
1.1	What is Needed . . . . .	5
1.2	Generating a Protein Structure File (PSF) . . . . .	5
1.3	Solvating the Protein . . . . .	9
1.3.1	Ubiquitin in a Water Sphere . . . . .	9
1.3.2	Ubiquitin in a Water Box . . . . .	10
1.4	Ubiquitin in a Water Sphere: Simulation with Non-Periodic Boundary Conditions . . . . .	12
1.4.1	Configuration File . . . . .	13
1.4.2	Run your Simulation . . . . .	19
1.5	Ubiquitin in a Water Box: Simulation with Periodic Boundary Conditions . . . . .	19
1.5.1	Configuration File . . . . .	20
1.5.2	Run your Simulation . . . . .	22
1.6	Output: Water Sphere Log File . . . . .	22
1.7	Analysis of Water Sphere Equilibration . . . . .	25
1.7.1	RMSD for Entire Protein . . . . .	25
1.7.2	RMSD for Protein without Last 5 Residues . . . . .	28
<b>2</b>	<b>Analysis</b>	<b>30</b>
2.1	Equilibrium . . . . .	30
2.1.1	RMSD for individual residues . . . . .	30
2.1.2	Maxwell-Boltzmann Energy Distribution . . . . .	33
2.1.3	Energies . . . . .	37
2.1.4	Temperature distribution . . . . .	39
2.1.5	Specific Heat . . . . .	42
2.2	Non-equilibrium properties of protein . . . . .	45
2.2.1	Heat Diffusion . . . . .	45
2.2.2	Temperature echoes . . . . .	49
<b>3</b>	<b>Steered Molecular Dynamics</b>	<b>59</b>
3.1	Removing Water Molecules . . . . .	59
3.2	Constant Velocity Pulling . . . . .	60
3.2.1	Fixed and SMD Atoms . . . . .	61
3.2.2	Configuration File . . . . .	63
3.2.3	Running the First SMD Simulation . . . . .	66
3.3	Constant Force Pulling . . . . .	67
3.3.1	The SMD Atom . . . . .	67
3.3.2	Configuration File . . . . .	68
3.3.3	Running the Second SMD Simulation . . . . .	69
3.4	Analysis of Results . . . . .	70
3.4.1	Force Analysis for Constant Velocity Pulling . . . . .	70
3.4.2	Distance Analysis for Constant Force Pulling . . . . .	72

<b>A PDB Files</b>	<b>75</b>
<b>B PSF Files</b>	<b>76</b>
<b>C Topology Files</b>	<b>78</b>
<b>D Parameter Files</b>	<b>85</b>
<b>E NAMD Configuration Files</b>	<b>91</b>
<b>F NAMD Standard Output</b>	<b>96</b>
<b>G Water Sphere tcl Script</b>	<b>99</b>

## Introduction

This tutorial provides a first introduction to NAMD and its basic capabilities. It can also be used as a refresher course for the non-expert NAMD user.

The tutorial assumes that you already have a working knowledge of VMD and that NAMD has been correctly installed on your computer. For installation instructions, please refer to the NAMD Users's Guide. For the accompanying VMD tutorial go to

<http://www.ks.uiuc.edu/Training/SumSchool03/Tutorials/vmd>.

The tutorial is subdivided in three sections. The first one covers the basic initial steps of a molecular dynamics simulation, i.e., minimization and equilibration of your system. The second section introduces typical simulation techniques and the analysis of equilibrium properties. The last section deals with *Steered Molecular Dynamics* and the analysis of unfolding pathways of proteins. Finally, brief descriptions of all files needed for the simulations are provided in the appendices.

For a detailed description of NAMD the reader is referred to the NAMD User's guide located at

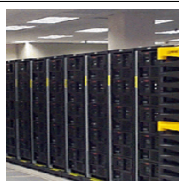
<http://www.ks.uiuc.edu/Research/namd/current/ug/>

The examples in the tutorial will focus on the study of ubiquitin – a small protein with interesting properties. Throughout the text, some material will be presented in separate “boxes”. Some of these boxes include complementary information to the tutorial, such as information about the biological role of ubiquitin, and tips or shortcuts for using NAMD. These boxes are not required for understanding the tutorial and may be skipped if you are short on time.

Boxes with an exclamation sign are specially important and should not be skipped.



**Warning!** The goal of this tutorial is to introduce NAMD by performing some short molecular dynamics simulations. Therefore, the examples provided are optimized so simulations can be done in a reasonable period of time on a common computing facility. This means that some parameters and conditions under which simulations are done in this tutorial are not suitable for scientific studies. Whenever this happens it will be pointed out and alternatives or more appropriate parameters/conditions will be provided in case you want to improve the simulations and/or you have more computer power available.



**Computer Related Material.** To aid in completing this tutorial, a web page of basic UNIX commands has been made available at <http://www.ks.uiuc.edu/Group/Events/SummerSchool/2003/handouts/unix.html>.

## First time

Begin by doing a copy of the files needed for the tutorial in a directory called `tbss.work` in your home directory. In a terminal window, move to this directory by typing:

```
tbss>cd ~/tbss.work
```

Copy the needed directory, but instead of typing `TOP_DIR`, type the location of the Summer School directory tree:

```
tbss>cp -r TOP_DIR/sumschool03/tutorials/02-namd-tutorial/namd-tutorial-files/ ./namd-tutorial-files
```

For instance, if the materials are located at `/mnt/cdrom` or at `~/Desktop`, replace `TOP_DIR` by `/mnt/cdrom` or `~/Desktop`. Use this copy of the tutorial files.

## Continue using the tutorial

If you logged out, and want to continue the tutorial, type in the terminal:

```
tbss> cd ~/tbss.work/namd-tutorial-files
```

This will place you in the directory containing all the necessary files.

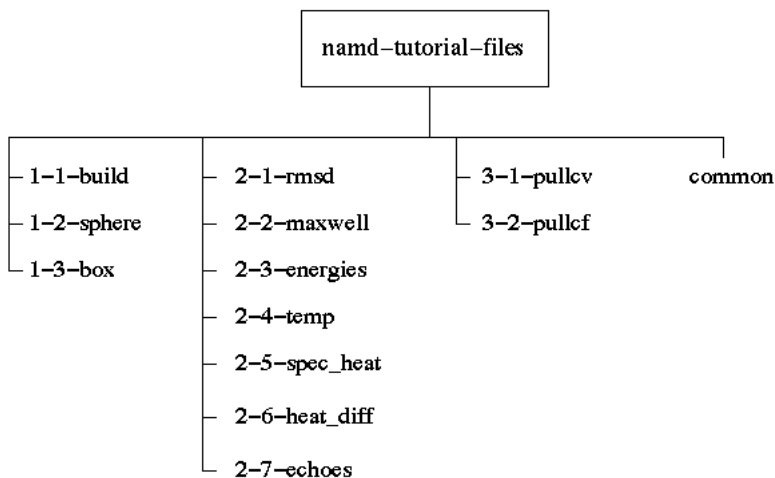


Figure 1: Directory Structure for tutorial exercises. Output for all simulations is provided in an “example-output” subdirectory within each folder shown.

# 1 Basics of NAMD

*In this section you will learn how to use NAMD to set up basic molecular dynamics (MD) simulations. You will learn about typical NAMD input and output files, in particular, those for protein energy minimization and equilibration in water.*

NOTE: You will be generating output data in this section by performing simulations and using other features of NAMD. These files are needed for Units 2 and 3. If you are not able to produce the output, correct versions have been provided for each section and may be found in the `example-output` folder inside the directory corresponding to the given simulation or exercise.

## 1.1 What is Needed

In order to run any MD simulation, NAMD requires at least four things:

- a PDB, or Protein Data Bank, file which stores atomic coordinates and/or velocities for the system. PDB files may be generated by hand, but they are also available via the Internet for many proteins at <http://www.pdb.org>. More in Appendix A.
- a PSF, or Protein Structure file, which stores structural information of the protein, such as various types of bonding interactions. More in Appendix B.
- a force field parameter file. A force field is a mathematical expression of the potential which atoms in the system feel. CHARMM, X-PLOR, AMBER, and GROMACS are four types of force fields, and NAMD is able to use all of them. The parameter file sets bond strengths, equilibrium lengths, etc. More in Appendix D.
- a configuration file, in which the user sets all the options that NAMD should adopt in running a simulation. The configuration file tells NAMD how the simulation is to be run. More in Appendix E.



**Force Field Topology File.** Later, you will make a psf file for your system. In doing so, a force field topology file is necessary. This file contains information on atom types, charges, and how the atoms are connected in a molecule. Note that the pdb file contains only coordinates, but not connectivity information! More in Appendix C.

## 1.2 Generating a Protein Structure File (PSF)

Of the four files mentioned above, an initial PDB file will typically be obtained through the Protein Data Bank, and the parameter and topology files for a given class of molecule may be obtained via the Internet at

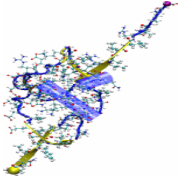
<http://www.pharmacy.umaryland.edu/~alex/research.html>. The NAMD configuration file is created by the user. The PSF file, however, must be *created* by the user from the initial PDB and parameter files.

- 1 Go to the `1-1-build` directory. In a Unix Terminal window, you can change directories using the `cd` command. Type `cd 1-1-build`. You can see the contents of the directory you are in by typing `ls`. In this folder, you will find many files that you will use later.

First, you will remove the water molecules from `1UBQ.pdb`, and create a clean `pdb` file of the protein alone.

- 2 Open VMD by typing `vmd` in the Unix Terminal window, and load `1UBQ.pdb` by clicking `File` → `New Molecule...` menu item in the VMD Main window. In the Molecule File Browser use the `Browse...` button to find the file `1UBQ.pdb`. Load it by pressing the `Load` button.

Note that the x-ray structure from the Protein Data Bank does not contain the hydrogen atoms of ubiquitin. This is because x-ray crystallography usually cannot resolve hydrogen atoms. The `pdb` file you generate along with the `psf` will contain guessed coordinates for hydrogen atoms of the structure. Later, energy minimization of the protein will ensure their positions are reasonable.



**X-ray Crystallography.** X-ray crystallography methods utilize the optical rule that electromagnetic radiation will interact most strongly with matter the dimensions of which are close to the wavelength of that radiation. X-rays are diffracted by the electron clouds in molecules, since both the wavelength of the x-rays and the diameter of the cloud are on the order of Angstroms. The diffraction patterns formed when a group of molecules is arranged in a regular, crystalline array, may be used to reconstruct a 3-D image of the molecule. Hydrogen atoms, however, are not typically detected by x-ray crystallography since their sizes are too small to interact with the radiation and since they contain only a single electron. The best x-ray crystallography resolutions currently available are around  $0.9\text{\AA}$ .

- 3 Choose the `Extensions` → `tkcon` menu item and in the VMD TkCon window, type the following commands:

```
set ubq [atomselect top protein]
$ubq writepdb ubqp.pdb
```

(Hit the Return key after each command.)

In the previous step you have created the file `ubqp.pdb`, which contains the coordinates of the ubiquitin alone without hydrogens, in the `1-1-build` directory.

- 4 Delete the current displayed molecule in VMD using the VMD Main window by clicking on `1UBQ.pdb` in the window, and selecting `Molecule` → `Delete Molecule`. Close VMD by typing `exit` in the `vmd console` window.
- 5 Now, you will create the psf file of ubiquitin. The `psfgen` package of VMD is very useful in this regard. In order to create a psf, you will first make a pgn file, which will be the target of `psfgen`. In a Unix Terminal window type `nedit ubq.pgn` to create a new file called `ubq.pgn`. (Be sure that you are in the `1-1-build` directory.) Then, type the following lines:

```
package require psfgen
topology top_all127_prot_lipid.inp
alias residue HIS HSE
alias atom ILE CD1 CD
segment U {pdb ubq.pdb}
coordpdb ubq.pdb U
guesscoord
writepdb ubq.pdb
writepsf ubq.psf
```



**Text Editing.** Nedit is a text editor which will be used throughout this tutorial to view and edit some of the files associated with the simulations. There are others such as `pico`, `emacs`, `jot`, and `vi`. Feel free to use whichever text editor you are most comfortable with.

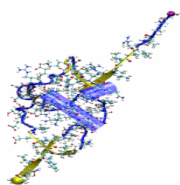
- 6 After typing this, save the file by clicking `File` → `Save` and quit the text editor by clicking `File` → `Exit`.

The file you just created contains the necessary commands to create the psf file of ubiquitin with hydrogen atoms and without water. Each command of the pgn file is explained:

- Line 1: You will be running `psfgen` within VMD. This line requires that the `psfgen` package is available to be called by VMD.
- Line 2: Load the topology file `top_all127_prot_lipid.inp`
- Line 3: Change the residue name of histidine to the proper name found in the topology file. `HSE` is one of three names for histidine, based on the protonation state of its side group. See the science box below for more information.
- Line 4: The atom named “`CD1`” ( $\delta$  carbon) in isoleucine residues is renamed as “`CD`”, its proper name from the topology file. Since isoleucine contains only one  $\delta$  carbon atom, the psf file does not use the number label after “`CD`”.



- Line 5: A segment called U is created, containing all atoms from `ubqp.pdb`. The `segment` command also adds hydrogen atoms.
- Line 6: Coordinates are read from `ubqp.pdb` and residue and atom names are matched. Old segment labels will be overridden with the new segment label “U”.
- Line 7: Coordinates of missing atoms (like hydrogens) are guessed based on residue definitions from the topology file.
- Line 8: A new pdb file with the complete coordinates of all atoms, including hydrogens, is written.
- Line 9: A psf file with the complete structural information of the protein is written.



**Histidine Residues.** Of the 20 amino acids, histidine is the only one which ionizes within the physiological pH range ( $\sim 7.4$ ). This effect is characterized by the  $pK_a$  of the amino acid side chain. For histidine, the value is 6.04. This leads to the possibility of different protonation states for histidine residues in a protein, and makes the consideration of the proper state important in MD simulations. The viable states are one in which the  $\delta$  nitrogen of histidine is protonated (listed with residue name “HSD” in the topology file), one in which the  $\epsilon$  nitrogen of histidine is protonated (“HSE”), and one in which both nitrogens are protonated (“HSP”).

- 7 In a Unix Terminal window (again, be sure that you are in the `1-1-build` directory), type the following command: `vmd -dispdev text -e ubq.pgn`. This will run the package `psfgen` on the file `ubq.pgn` in the text only version of VMD (no graphics). It will generate the psf and the pdb file of ubiquitin with hydrogens.

In your screen you will see different messages. Warnings are related to the ends of your molecule and are normal. Your system should have 1231 atoms and 631 with guessed coordinates.

- 8 Close the text version of VMD by typing `exit` at the `vmd >` prompt.

Having run `psfgen`, two new files will now appear in your `1-1-build` directory: `ubq.pdb` and `ubq.psf`. Check this by typing `ls`. You have created the psf! You may want to inspect `ubq.pdb` using `nedit`.



**Multiple Chains.** Often, pdb files will contain structural information for polymers of proteins. Each monomer is typically labeled as a separate chain in the pdb file. Chain ID is indicated in the column of the pdb after the residue name. In those cases, you can use VMD to make separate pdb files for each chain, using the following commands in the VMD TkCon window:

```
set chainA [atomselect top "chain A"]  
$chainA writepdb chainA.pdb
```

These commands should be used for each chain in the system. The psfgen package requires a pdb file for each chain.

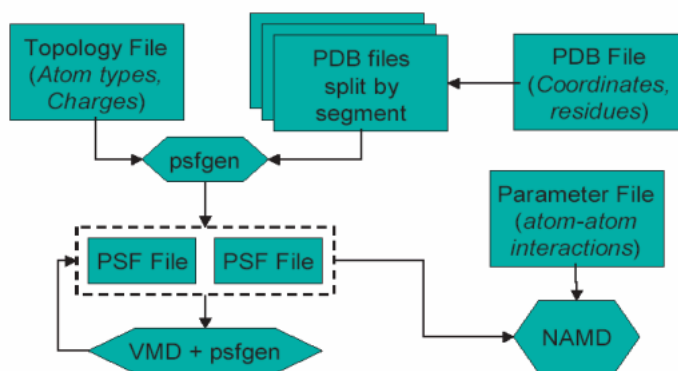


Figure 2: Flowchart indicating the role of files as used by VMD, NAMD, and psfgen.

## 1.3 Solvating the Protein

Now, the protein needs to be solvated, i.e., put inside water, to more closely resemble the cellular environment. You will do so in two ways, placing ubiquitin in:

- a water sphere in vacuum, in preparation for minimization and equilibration without periodic boundary conditions.
- a water box, in preparation for minimization and equilibration with periodic boundary conditions.

### 1.3.1 Ubiquitin in a Water Sphere

You will utilize a prepared tcl script to create the water sphere. It is called `wsphere.tcl` and is located in your `1-1-build` directory. The syntax of the

script itself is given in Appendix G.

- 1 In a Unix Terminal window in your 1-1-build directory, type `vmd -dispdev text -e wsphere.tcl`. This will call the script, which will place ubiquitin in the smallest possible water sphere which completely immerses the protein.



**Water Sphere.** The creation of a water sphere is not a necessary step if one wishes to examine a protein in water surrounded by vacuum. However, it is a sensible one. If the water box is left as is, minimization and equilibration will simply lead the box to deform into the most stable shape possible, minimizing the surface tension. That shape is a sphere! Creating the sphere before beginning simulations reduces the amount of time needed to achieve equilibration and saves computational effort.

- 2 The output of the `wsphere.tcl` file will be the center of mass and radius of the sphere. Record these numbers. The script will have created the `pdb` and `psf` files `ubq_ws.pdb` and `ubq_ws.psf`, respectively, of ubiquitin in a water sphere. They will be in your 1-1-build directory. Check this by typing `ls`.

### 1.3.2 Ubiquitin in a Water Box

- 1 Launch the graphical version of VMD by typing `vmd` in the Unix Terminal window.
- 2 In the VMD Main window, click Extensions → tkcon. In the VMD TkCon window of your VMD session type:

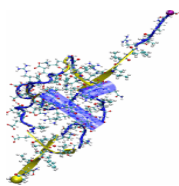
```
package require solvate
solvate ubq.psf ubq.pdb -t 5 -o ubq_wb
```

The `package require solvate` command simply places the `solvate` package in the correct place, so that VMD will be able to call it.

The `solvate` package will put your protein (described in `ubq.psf` and `ubq.pdb`) in a box of water. The `-t` option creates the water box dimensions such that there is a layer of water 5Å in each direction from the atom with the largest coordinate in that direction. The `-o` option creates the output files `ubq_wb.pdb` and `ubq_wb.psf` for ubiquitin with the water box. You will also obtain the files `combine.pdb` and `combine.psf` which may be eliminated. They are intermediate files used by the `solvate` program to transport information into VMD.



**Size of Water Box.** The water box created is somewhat small for this study, especially considering that you will pull and unravel the protein in Unit 3 of this tutorial. Ideally, one should work with a box which is large enough that the protein does not interact with its image in the next cell if periodic boundary conditions are used. Periodic boundary conditions will be explained in section 1.5. Furthermore, if the protein is being pulled, the box should be large enough that the water will still significantly immerse the protein when it is fully extended.



**Ions.** Ions should be placed in the water to represent a more typical biological environment. They are especially necessary if the protein being studied carries an excess charge. In that case, the number of ions should be chosen to make the system neutral. The ions present will shield the regions of the protein which carry the charge, and make the entire system more stable. They should be placed in regions of potential minima, since they will be forced to those regions during the simulation anyway. The psf file contains the charge of each atom and may be used to determine the charge of the total system or parts of it.

- 3 In the VMD Main window, click File → New Molecule. In the Molecule File Browser use the Browse... button to find the file `ubq_wb.pdb`. Load it by pressing the Load button.
- 4 In the same Molecule File Browser, browse for `ubq_wb.psf` and load it. This will load the structural information into VMD. In the display, check if the water box surrounds ubiquitin, and demonstrate to yourself that the hydrogen atoms now exist.
- 5 In the VMD TkCon window type:

```
set everyone [atomselect top all]
measure minmax $everyone
```

This analyzes all atoms in the system and gives you the minimum and maximum values of  $x$ ,  $y$  and  $z$  coordinates of the entire protein-water system.
- 6 These values are defined relative to the origin of the coordinate system, set by the initial pdb file, `1UBQ.pdb`. The center of the water box may be determined by calculating the midpoint of each of the three sides in the coordinate system. For example, if minimum and maximum values of  $x$  are returned as 10.44 and 51.12, respectively,  $x$ -coordinate of the center of the box would be  $\frac{51.12+10.44}{2} = 30.78$ . Alternatively, one can use tcl scripting in VMD to find the center of the box by typing `measure center $everyone` in the VMD TkCon window. Determine the coordinates of the center of your water box and record these numbers.
- 7 Close VMD by typing `exit` in the vmd console window.

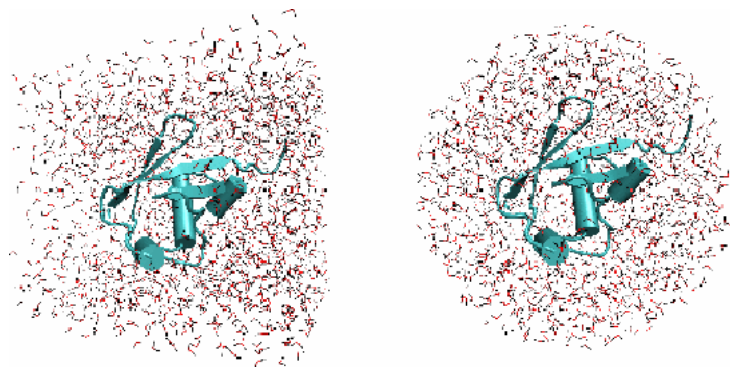
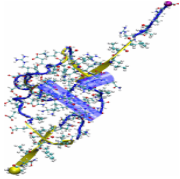


Figure 3: Ubiquitin in a water box and in a water sphere. Hydrogen atoms are colored black for contrast.

Now that you have the pdb and psf files for solvated ubiquitin, along with the parameter file for proteins in general, you only need a NAMD configuration file to be able to minimize and equilibrate your ubiquitin-water system. In the next two sections, you will edit a NAMD configuration file provided to you in order to gain a beginner's understanding of its function.



**Minimization and Equilibration.** Minimization and equilibration differ from each other by the nature in which they implement the supplied force field. Energy minimization involves searching the energy landscape of the molecule for minima, i.e., places in which the molecule is relaxed, by systematically varying the positions of atoms and calculating the energy. Equilibration involves molecular dynamics, whereby Newton's laws are solved for each atom in the system to dictate their trajectories. Achievement of equilibrium is judged by how well velocities, pressure, etc. are distributed in the system over a given amount of time.

8 Before beginning the next section, you must copy the necessary pdb and psf files into your `common` folder. In a Unix Terminal window, type `cp ubq.pdb ubq.psf ubq.w* ../common` and hit `Enter`. Then, type `ls ../common` to see which files you have copied.

## 1.4 Ubiquitin in a Water Sphere: Simulation with Non-Periodic Boundary Conditions

In this section, you will examine the minimization and equilibration of ubiquitin in a water sphere placed in vacuum.

### 1.4.1 Configuration File



**Configuration Warning.** Be very careful when typing commands into your configuration files. Even the slightest typographical error will cause the simulation not to run. The point is not simply to run a simulation, but to run it correctly. Be sure your file is correct before you submit a simulation to be run!



**Configuration File Explanation.** This section will introduce a configuration file to you. The file is complete and ready for submission *as is*. Since you will be examining the file line by line in the section, you may skip to section 1.4.2 if you are very short on time, and examine the contents of this section later.

- 1 Go to your `1-2-sphere` directory by typing `cd ../1-2-sphere`. Here, you will find a configuration file for the minimization and equilibration of ubiquitin in a water sphere.

All output files for the minimization and equilibration of your ubiquitin in a water sphere system will be placed in this directory. The configuration file is the only input file placed here, since it is particular to this simulation. The `pdb`, `psf`, and parameter files, which may be used by many other simulations, are placed in your `common` directory and are *called* by each respective configuration file.

- 2 Open the configuration file, `ubq_ws_eq.conf` by typing `nedit ubq_ws_eq.conf`.

The configuration file may seem complex at first, but it will be examined line by line to determine its function in your simulation.

Note that when “`#`” appears at the beginning of a line, the entire line is not recognized as compilable script. These lines are “commented out”, rather. In the middle of a line, “`;``#`” is used to comment out the remainder of the line.

- 3 The “Job Description” section contains only comments, and its purpose is to inform those who view the configuration file about what it is meant for. Your comment should read

```
# Minimization and Equilibration of  
# Ubiquitin in a Water Sphere
```

- 4 The “Adjustable Parameters” section contains five commands:

- **structure:** calls the `psf` file describing the system (`../common/ubq_ws.psf`).
- **coordinates:** calls the initial coordinate data from the file listed next to it (in this case `../common/ubq_ws.pdb`).

- **set temperature**: creates a variable called “temperature” in which to store a value for the initial temperature of the system. If you place the text “**\$temperature**” in the configuration file, NAMD simply reads it as a label for the number “310”. (Creating variables is useful since you can alter the value of that variable in a single place if it is listed many times in the configuration file.)
- **set outputname**: creates a variable called “outputname” in which to store a generic name for output files. If you place the text “**\$outputname**” in the configuration file, NAMD simply reads it as a label for “ubq.ws.eq”.
- **firsttimestep**: simply sets a number value for the first time step of the simulation. It is typically useful when restarting a simulation. For instance, if a previous simulation ended at time step 552, the command **firsttimestep 553** would be used.

5 The “Simulation Parameters” section contains many commands, commented into different categories:

- **Input**
  - **paraTypeCharmm**: indicates whether or not the parameter file is in the format used by the CHARMM force field. **on** indicates that it is; **off** indicates that it is not. (If this command is not specified, NAMD assumes the file is in X-PLOR format by default.)
  - **parameters**: calls the force field parameters from the file listed next to it (in this case `../common/par_all27_prot_lipid.inp`).
  - **temperature**: sets the initial temperature of the system, in K, with the value listed next to it (in this case `$temperature`, or 310). This is done by assigning random velocities to atoms such that their average kinetic energy accurately represents the given temperature.
- **Force-Field Parameters**
  - **exclude**: specifies which atomic interactions are to be excluded from consideration. See Figure 4 for general atom labels. The **scaled1-4** value indicates that interactions between any such atoms 1 and 3 are neglected and interactions between atoms 1 and 4 are modified. The van der Waals interaction for “1-4” atoms are modified using special 1-4 parameters defined in the parameter files, and electrostatic interaction is modified as shown in the next command.
  - **1-4scaling**: specifies the degree to which the electrostatic interaction between 1-4 atoms is to be taken into account. It may be a decimal between 0 and 1 (here, it is 1) and indicates how much the interaction is “turned off” or “on”, respectively.

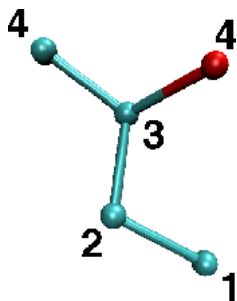


Figure 4: Number labels for atoms which are a given amount of bonds away from one another

- **cutoff**: indicates the distance in Å beyond which electrostatic and van der Waals interactions are cut-off. Otherwise, those interactions are considered over the entire volume of your system. This can be computationally too costly. The definition changes when the Particle Mesh Ewald Sum is invoked. See Section 1.5.1.
  - **switching**: indicates whether or not switching functions are used to smoothly take electrostatic and van der Waals interactions to zero at the **cutoff** distance. You may list “on” or “off” next to it as a yes/no answer.
  - **switchdist**: indicates the distance in Å at which the functional form of electrostatic and van der Waals interactions is modified to allow their values to approach zero at the **cutoff** distance. A visual explanation of this is useful and may be found in Figure 5.
  - **pairlistdist**: is designed to make computation faster. It specifies a distance in Å. NAMD will only search within this distance for atoms which may interact by electrostatic or van der Waals interactions. This way, NAMD does not have to search the entire system. The distance must be greater than the **cutoff** distance, and the list must be updated during the simulation. See Figure 5.
- **Integrator Parameters**
    - **timestep**: indicates the value of the time step size used in the simulation. MD simulations solve Newton’s laws to determine the trajectories of atoms. The time step tells NAMD how to discretize the particle dynamics. It is specified in femtoseconds (here, 2 fs).
    - **rigidBonds**: specifies which bonds involving hydrogen are considered to be rigid (non-vibrating). The value **all** specifies all linear bonds involving hydrogen and any other atoms.



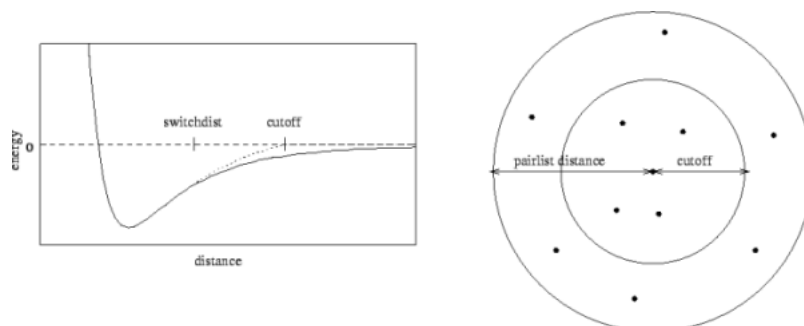


Figure 5: Cutoff and switching distances indicated on the left. Pair list distance indicated on the right.

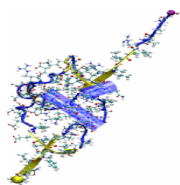
- **nonbondedFreq**: specifies in number of time steps how often nonbonded interactions should be calculated. It is useful for saving computational time.
- **fullElectFrequency**: specifies in number of time steps how often full electrostatic interactions should be calculated.
- **stepspercycle**: Atoms are reassigned pair list identities (as explained above) once every cycle. This command specifies how long one cycle lasts, i.e. the number of time steps in one cycle.
- Constant Temperature Control
  - **langevin**: indicates whether or not the simulation uses Langevin dynamics; uses values **on** and **off**. See the science box below for more on Langevin dynamics.
  - **langevinDamping**: sets the value of the Langevin coupling coefficient, which quantifies the friction applied to the system, removing energy from the system, slowing atoms down, etc. It is specified in  $\text{ps}^{-1}$ .
  - **langevinTemp**: Langevin dynamics may be applied to all atoms or only non-hydrogen atoms in the system. This command specifies the temperature at which to keep those atoms, even though friction and random forces will be acting on them. (Remember **\$temperature** is a variable for the value 310.)
  - **langevinHydrogen**: indicates whether or not Langevin dynamics will be applied to hydrogen atoms in the simulation; uses values **on** and **off**.
- Output
  - **outputName**: Several types of output data may be written by NAMD for any simulation. This command specifies the prefix for output filenames. NAMD always returns the following two output files for *every* simulation: a pdb file containing the final

coordinates of all atoms in the system and a pdb file containing the final velocities of all atoms in the system; the extensions for these files are `.coor` and `.vel`, respectively. Thus, this configuration file will return two files named `ubq_ws_eq.coor` and `ubq_ws_eq.vel`

- `restartfreq`: During the simulation, NAMD can also create restart files, one of which is a pdb file which stores atomic coordinates, and the other of which stores atomic velocities. This command specifies the amount of time steps between writing to the restart file (here, every 500 steps, or 1000fs, or 1 ps). If this command is not set, NAMD will not create a restart file. Furthermore, NAMD will store the file from the previous cycle each time it writes a new file. The filename is appended with a `.old` extension; it is created in case NAMD fails in writing the new restart file.
- `dcdfreq`: The dcd file contains only atomic coordinates, and they are written to the file several times over the course of a simulation. Thus, it provides a *trajectory* of the system over the runtime. This command specifies the number of time steps between writing new coordinates to the dcd output file. If this command is not set, NAMD will not create a dcd file.

In addition to the output files described, NAMD also returned a log of the simulation. This file is explained further below.

- `outputEnergies`: specifies the number of time steps between each output of system energies (for various force field interactions) into the `.log` file (here, every 100 steps, or 200 fs).
- `outputPressure`: specifies the number of time steps between each output of system pressure into the `.log` file.



**Langevin Dynamics.** Langevin dynamics is a means of controlling the kinetic energy of the system, and thus, controlling the system temperature and/or pressure. The method uses the Langevin equation for a single particle:

$$m_i \frac{d^2 x_i(t)}{dt^2} = \mathbf{F}_i\{x_i(t)\} - \gamma_i \frac{dx_i(t)}{dt} m_i + \mathbf{R}_i(t)$$

Here, two additional terms on the right hand side accompany the ordinary force the particle experiences. The second term represents a frictional damping that is applied to the particle with frictional coefficient  $\gamma_i m_i$ . The third term represents random forces which may be applied to the particle (as a result of solvent interaction). These two terms are used to maintain particle kinetic energy to keep system temperature, for instance, at a constant value.



**Rigid Bonds.** The time step used in any MD simulation should be dictated by the fastest process (i.e. movement of atoms) taking place in the system. Among the various interactions, bond stretching and angle bending are the fastest, with typical bond stretching vibrations occurring on the order of once every 10-100 femtoseconds. Using a time step of 2 fs, which is close to the vibrational period (10 fs) of linear bonds involving hydrogen (shortest vibration, since it has small mass), requires that these bonds be fixed, and only slower vibrations may be free to move, such as dihedral angle bending. For large molecules, these slower vibrations typically govern the behavior of the molecule more so than the quicker ones, so bond fixing is somewhat acceptable, but should be avoided for accurate simulations.

6 The “Extra Parameters” section contains commands which are applicable to more specific simulations. Here have been placed commands which characterize the spherical boundary conditions on the water sphere. These conditions prevent the sphere from undergoing evaporation or diffusion.

- Spherical Boundary Conditions
  - **sphericalBC**: indicates whether or not the simulation uses spherical boundary conditions; uses values **on** and **off**.
  - **sphericalBCcenter**: sets the  $x$ ,  $y$ ,  $z$  coordinates of the center of the sphere to which spherical boundary conditions are applied. This is why you need to record the center of your water sphere after you solvate!

The next three commands are parameters which must be specified for spherical boundary conditions to work properly.

- **sphericalBCr1**: sets the distance in Å at which the first boundary potential begins to act.
- **sphericalBCk1**: The first potential applied by spherical boundary conditions to keep the sphere together (or pull it apart) is a harmonic one. This command sets the value of the force constant for that potential. Its value is specified in kcal/mol·Å<sup>2</sup>.
- **sphericalBCexp1**: sets the value of the exponent used to formulate the potential; must be a positive, even integer.

7 The “Execution Script” section contains three commands, the first two of which apply to minimization and the last one of which applies to equilibration.

- Minimization
  - **minimize**: sets the number of iterations over which to vary atom positions to search for a minimum in the potential (in this case 100).

- **reinitvels**: Minimization is performed on the system after all atomic velocities have been set to zero. This command resets atomic velocities such that the system starts at the temperature specified (in this case, **\$temperature**, or 310K).
- **run**: sets the number of time steps over which to run the MD equilibration (in this case 5000, which corresponds to 10,000 fs or 10 ps, since a 2 fs time step has been used).

8 Now, close the configuration file by clicking File → Exit.



**NAMD User's Guide.** The explanations of configuration file commands in the previous sections is far from complete. For an explanation of a more general configuration file, see Appendix E. For a more in-depth explanation of configuration file commands and for acceptable values which may be supplied with those commands, see the NAMD User's Guide at <http://www.ks.uiuc.edu/Research/namd/current/ug/>.



**Required Configuration File Parameters.** The following parameters are required by NAMD to appear in every configuration file: `coordinates`, `structure`, `parameters`, `exclude`, `outputname`, `numsteps` and one of the following: `temperature`, `velocities`, or `binvelocities`.

#### 1.4.2 Run your Simulation

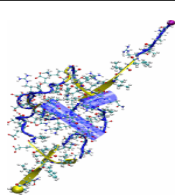


**Running.** Run your simulation using the configuration file `ubq_ws_eq.conf` and the label `ubq_ws_eq.log` for your output log file. See handout for further instructions.

Your simulation should take about 20 minutes to complete, and NAMD will produce the output files described in Section 1.6.

### 1.5 Ubiquitin in a Water Box: Simulation with Periodic Boundary Conditions

In this section, you will examine the minimization and equilibration of ubiquitin in a water box with periodic boundary conditions.



**Periodic Boundary Conditions.** Periodic boundary conditions involve surrounding the system under study with identical virtual unit cells. The surrounding virtual systems interact with atoms in the real system, creating an environment in which the system effectively sees no vacuum. These modeling conditions are effective in eliminating surface interaction of the water molecules which make the system somewhat different than an *in vivo* environment.

### 1.5.1 Configuration File

- 1 Go to your 1-3-box directory by typing `cd ../1-3-box`. Here, you will find a configuration file for the minimization and equilibration of ubiquitin in a water box. All output files for the minimization and equilibration of your ubiquitin in a water box system will be placed in this directory.
- 2 Open the configuration file, `ubq_wb_eq.conf` by typing `nedit ubq_wb_eq.conf`.

The configuration file contains some commands which are different than the water sphere configuration file. Here, these differences are pointed out and explained.

- 3 The only differences in this file lie in the “Simulation Parameters” section, where three new categories of parameters have been added. The “Output” section has also been modified. The new commands are listed:
  - Periodic Boundary Conditions
    - Three periodic cell basis vectors are to be specified to give the periodic cell its shape and size. They are `cellBasisVector1`, `cellBasisVector2`, and `cellBasisVector3`. In this file, each vector is perpendicular to the other two, as indicated by a single  $x$ ,  $y$ , or  $z$  value being specified by each. For instance, `cellBasisVector1` is  $x = 42\text{\AA}$ ,  $y = 0\text{\AA}$ ,  $z = 0\text{\AA}$ . With each vector perpendicular, a rectangular 3-D box is formed.
    - `cellOrigin`: specifies the coordinates of the center of the periodic cell in  $\text{\AA}$ . This is why you need to calculate the center of your water box after you solvate!
    - `wrapWater`: This command may be used with periodic boundary conditions. If a water molecule crosses the periodic boundary, leaving the cell, setting this command to `on` will translate its coordinates to the mirror point on the opposite side of the cell. Nothing can escape. The command may be set to `on` or `off`.
    - `wrapAll`: same as `wrapWater`, except this applies to all molecules.
  - PME (for full-system periodic electrostatics)

PME stands for Particle Mesh Ewald Sum, and is useful in dealing with potentials in the system when periodic boundary conditions are

present. The Ewald sum is an efficient way of calculating long range forces in the periodic system. The particle mesh is a 3-D grid created in the system over which the system charge is distributed. From this charge, potentials and forces on atoms in the system are determined.

- **PME**: indicates whether or not the simulation uses the Particle Mesh Ewald Sum method; uses values **yes** and **no**.
- **PMEGridSizeX**: sets the size of the PME grid in the direction *along cellBasisVector1* (not the  $x$  direction as implied); the number of grid points in the system in the direction specified by **cellBasisVector1**.
- **PMEGridSizeY**: sets the size of the PME grid in the **cellBasisVector2** direction.
- **PMEGridSizeZ**: sets the size of the PME grid in the **cellBasisVector3** direction.

Note that since **cellBasisVector** is defined with slightly different values in each direction, the size of the mesh spacing (in length) will be different in each direction.

Note also that when using the PME method, the command **cutoff** dictates the separation between long and short range forces for the method; it does not simply turn off interactions.

- **Constant Pressure Control (variable volume)**
  - **useGroupPressure**: NAMD calculates system pressure based on the forces between atoms and their kinetic energies. This command specifies whether interactions involving hydrogen should be counted for all hydrogen atoms or simply between groups of hydrogen atoms; uses values **yes** and **no** and must be set to **yes** if **rigidBonds** are set.
  - **useFlexibleCell**: specifies whether or not you want to allow the three dimensions of the periodic cell to vary *independently*; uses values **yes** and **no**.
  - **useConstantArea**: NAMD allows you to keep the  $x - y$  cross sectional area constant while varying the  $z$  dimension; uses values **yes** and **no**.
  - **langevinPiston**: indicates whether or not the simulation uses a Langevin piston to control the system pressure; uses values **on** and **off**.
  - **langevinPistonTarget**: specifies, in units of bars, the pressure which the Langevin piston tries to maintain. (1 atm = 1.013 bars)  
The following are specifications for the Langevin piston which NAMD allows you to specify.

- `langevinPistonPeriod`: sets the oscillation time scale in fs for the Langevin piston.
- `langevinPistonDecay`: sets the damping time scale in fs for the Langevin piston.
- `langevinPistonTemp`: sets the “noise” temperature in K for the Langevin piston; should be set equal to the target temperature for the temperature control method (here, set by `langevinTemp`).

- Output

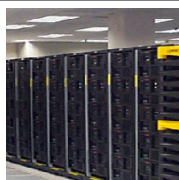
- `xstFreq`: The *extended system trajectory* file contains a record of the periodic cell parameters, essentially recording the trajectory of the cell boundaries over the run time. This command specifies how often, in time steps, the configuration will be recorded. If this command is set, three xst files will be output: 1 final and 2 restarts.

Note that the commands which specify spherical boundary conditions have been completely removed, since this simulation is using periodic boundary conditions.



**Minimization and Equilibration.** Typically, MD minimization and equilibration simulations involve more than one minimization-equilibration cycle, often fixing and releasing molecules in the system. For instance, one typically minimizes the system and then equilibrates with the atoms in the protein fixed in space, and then minimizes the system again and equilibrates again, this time with the protein free to move. Fixing the protein allows the water, which typically responds much faster to forces than the protein, to do the relaxing in the first step, saving computational effort.

### 1.5.2 Run your Simulation



**Running.** Run your simulation using the configuration file `ubq_wb_eq.conf` and the label `ubq_wb_eq.log` for your output log file. See handout for further instructions.

## 1.6 Output: Water Sphere Log File

- 1 Output of your water sphere minimization-equilibration simulation will be returned to the `1-2-sphere` directory. Go there by typing `cd ../1-2-sphere`. List the files in the directory by typing `ls`.

Based on the descriptions in the Output part of the “Simulation Parameters” section, your present simulation should yield these eleven output files:

- `ubq_ws_eq.log`
- `ubq_ws_eq.coor`
- `ubq_ws_eq.vel`
- `ubq_ws_eq.xsc`
- `ubq_ws_eq.dcd`
- `ubq_ws_eq.restart.coor`
- `ubq_ws_eq.restart.vel`
- `ubq_ws_eq.restart.xsc`
- `ubq_ws_eq.restart.coor.old`
- `ubq_ws_eq.restart.vel.old`
- `ubq_ws_eq.restart.xsc.old`

(Your original configuration file should also still be in the directory.)

Seven of the output files are in binary code so that coordinate and velocity information may be kept to high precision. The files which are not in binary code are:

- \* `ubq_ws_eq.xsc`, `ubq_ws_eq.restart.xsc`, `ubq_ws_eq.restart.xsc.old`, which are *extended system configuration* files. They store the periodic cell dimensions of the system and the time step at which this information was taken. They are not useful when periodic boundary conditions are not used. (The restart files function in the same way as described before.)
  - \* `ubq_ws_eq.log`, the log file. It is here that the energy of the various force field interactions is stored. The log file stores a lot of information and it will be examined.
- 2 The first section of the log file, indicated by the `Info` tag, contains the parameters used to run the simulation, as other useful information about the system, such as the number of atoms, the type and number of bonds, the total mass, and the total charge.
  - 3 After the first section, information about the minimization of the system is displayed beginning with the line labeled `TCL`. This line states how long the minimization lasts.
  - 4 After that, the pressure of the system is given. For the present simulation, no pressure output is given because NAMD does not define the pressure for a non-periodic system. For your simulation using periodic boundary conditions, you will see values for the pressure.



- The next part gives a listing of energies of interaction, first defining which are listed with **ETITLE**, then listing the actual values in kcal/mol every time step next to **ENERGY**. (The time step (TS) value has no units, and the temperature (TEMP) value is in K.)
- The values listed for **INITIAL STEP** and **GRADIENT TOLERANCE** are mathematical parameters corresponding to the numerical search for a minimum in the potential. **GRADIENT TOLERANCE** will decrease over the course of the minimization and may be used to roughly judge the extent of minimization. See Figure 6.

```

Info: NONZERO IMPRECISION IN COULOMB TABLE: 1.7E-13 2.1E-13 1.2E-13 2.1E-13
)
Info: Entering startup phase 8 with 35208 kB of memory in use.
Info: Finished startup with 35208 kB of memory in use.
TCL: Minimizing for 100 steps
PRESSURE: 0 0 0 0 0 0 0 0 0 0
GPRESSURE: 0 0 0 0 0 0 0 0 0 0
ETITLE:      TS      BOND      ANGLE      DIHED      IMPRP  \
            ELECT      VDW      BOUNDARY    MISC      KINETIC \
            TOTAL      TEMP      TOTAL2     TOTAL3    TEMPAV\
G
ENERGY:      0      1913.4792    3008.0536    358.3486    10.7302  \
            -18281.1014    5217.3380    0.0000    0.0000    0.0000  \
            -7773.1518    0.0000    -7773.1518    -7773.1518    0.0000\
0
INITIAL STEP: 1e-06
GRADIENT TOLERANCE: 15332.5
PRESSURE: 1 0 0 0 0 0 0 0 0 0
GPRESSURE: 1 0 0 0 0 0 0 0 0 0

```

Figure 6: Typical minimization output of the log file

- After minimization, equilibration data are presented in the log file. See Figure 7. The first lines shown in the figure confirm that NAMD is writing the proper information at the proper time step, which you have specified in the configuration file. A warning about the size of the **pairlistdist** is followed by the space where pressures would normally be written. Note that since they are not, this space is used to store the total time step of the simulation. Energies at that time step are then fully reported. In this case, they are output every 100 time steps, as specified in the configuration file.
- The end of the file confirms that the final frame of the simulation is being recorded, and provides computational data which may be used to benchmark your run should you wish to change parameters, run on another computer, etc.

The log file for your simulation of ubiquitin in a water box is quite similar to the one described above with the exception of pressure output, as described.

```

5
-14344.0640      311.2559      -14327.2768      -14325.5022      311.677\
OPENING EXTENDED SYSTEM TRAJECTORY FILE
WRITING EXTENDED SYSTEM TO RESTART FILE AT STEP 500
OPENING COORDINATE DCD FILE
WRITING COORDINATES TO DCD FILE AT STEP 500
WRITING COORDINATES TO RESTART FILE AT STEP 500
FINISHED WRITING RESTART COORDINATES
WRITING VELOCITIES TO RESTART FILE AT STEP 500
FINISHED WRITING RESTART VELOCITIES
Warning: Pairlistdist is too small for 1 patches during timestep 507.
Warning: Pairlists partially disabled; reduced performance likely.
PRESSURE: 600 0 0 0 0 0 0 0 0
GPRESSURE: 600 0 0 0 0 0 0 0 0
Warning: 213 pairlist warnings since previous energy output.
ENERGY:      600      248.0650      696.2189      395.5456      43.5162  \
            -21726.6737      1608.2327      4.3657      0.0000      4274.6493  \
            -14456.0802      315.9203      -14435.9374      -14445.7993      313.365\
3

```

Figure 7: Typical equilibration output of the log file

See Appendix F for more information on log files.

## 1.7 Analysis of Water Sphere Equilibration

In this section, you will analyze the extent to which your system has equilibrated using what is known as the *Root Mean Square Deviation*, or RMSD. The RMSD characterizes the amount by which a given selection of your molecule deviates from a defined position in space. You will use the output files from your minimization and equilibration of ubiquitin in a water sphere to calculate RMSD values and analyze the extent of equilibration of the simulation. RMSD values will be calculated for all atoms of the protein backbone (without hydrogens) for the entire protein and for the protein excluding the last five residues.

### 1.7.1 RMSD for Entire Protein

Your minimization-equilibration simulation generated a trajectory for the system called `ubq_ws_eq.dcd`. That trajectory is key to calculating the RMSD of the protein over the course of the equilibration. The simulation was performed in a so-called *NVT ensemble* in which the number of particles,  $N$ , the volume,  $V$ , and the temperature,  $T$ , are kept constant.

NAMD has analysis tools available at [www.ks.uiuc.edu/Research/namd/utilities/](http://www.ks.uiuc.edu/Research/namd/utilities/). One of them is the script `namdplot`. This script gets the relevant data from the log files for values of energies, temperature, etc. over time and then uses the package `xmgrace` to plot them.

```
namdplot var1 [var2] [var...] file
- plots the thermodynamical variables var1, var2,...
from the NAMD output file file
```

- 1 In the Unix Terminal window, type:

```
namdplot TEMP ubq_ws_eq.log
```

This will get all the temperature data from the log file and plot it over time. `namdplot` uses `xmgrace` to plot, so you need to be sure it is installed.

- 2 To know which thermodynamic variables are available to plot from a log file, you can type:

```
namdplot ubq_ws_eq.log
```

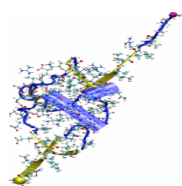
You will see something like:

```
ETITLE: TS BOND ANGLE DIHED IMPRP ELECT VDW BOUNDARY MISC
KINETIC TOTAL TEMP TOTAL2 TOTAL3 TEMPAVG
```

in the first line, where TS stands for time step, BOND for bond energy, and so forth.

- 3 Try using `namdplot` for TEMP (temperature) and TOTAL (total energy). You should see these variables converging to an average value for an equilibrated system.

The thermodynamic variables that you checked above tell you about the thermodynamic state of the whole system. But you would also like to know if your protein is conformationally stable. For this, you will calculate the RMSD of the protein backbone. This will give you an idea of the stability of the protein. If the RMSD is still increasing at the end of the run, it means your protein is still searching for a lower energy state, and thus is not yet equilibrated!



**Root Mean Square Deviation.** The Root Mean Squared Deviation (RMSD) is a numerical measure of the difference between two structures. It is defined as:

$$RMSD_{\alpha} = \sqrt{\frac{\sum_{j=1}^{N_t} \sum_{\alpha=1}^{N_{\alpha}} (\vec{r}_{\alpha}(t_j) - \langle \vec{r}_{\alpha} \rangle)^2}{N_{\alpha}}}$$

where  $N_{\alpha}$  is the number of atoms whose positions are being compared,  $N_t$  is the number of time stpes over which atomic positions are being compared,  $\vec{r}_{\alpha}(t_j)$  is the position of atom  $\alpha$  at time  $t_j$ , and  $\langle \vec{r}_{\alpha} \rangle$  is the average value of the position of atom  $\alpha$  to which the positions  $\vec{r}_{\alpha}(t_j)$  are being compared. It is defined as:

$$\langle \vec{r}_{\alpha} \rangle = \frac{1}{N_t} \sum_{j=1}^{N_t} \vec{r}_{\alpha}(t_j)$$

Note the script you are using takes the first time step of the simulation as a reference, rather than using an average value.

You will use VMD to calculate RMSD values:

- 4 Launch VMD by typing `vmd` in the Unix Terminal window.
- 5 Load your psf file for the water sphere, `ubq_ws.psf`, which is in your common directory, by clicking File → New Molecule... in the VMD Main window. Click the Browse button in the Molecule File Browser window. Click Up one directory in the Choose a molecule file window. Then click common and then `ubq_ws.psf` in the same window. Click OK and then click Load.
- 6 Click the Browse button in the Molecule File Browser window. Click Up one directory in the Choose a molecule file window. Then click 1-2-sphere and then `ubq_ws_eq.dcd` in the same window. Click OK and then click Load. You have loaded the trajectory of your equilibration.
- 7 The script we are going to use is called `rmsd.tcl`. This is the script content:

```
set outfile [open rmsd.dat w]
set nf [molinfo top get numframes]
set frame0 [atomselect top "protein and backbone and noh" frame 0]
# rmsd calculation loop
for { set i 1 } { $i ≤ $nf } { incr i } {
  set sel [atomselect top "protein and backbone and noh" frame $i]
  $sel move [measure fit $sel $frame0]
  puts $outfile "[measure rmsd $sel $frame0]"
}
close $outfile
```

- 8 The script does the following:

- Open file `rmsd.dat` for writing.  
`set outfile [open rmsd.dat w]`
- Get the number of frames in the trajectory and assign this value to the variable `nf`  
`set nf [molinfo top get numframes]`
- Select the first frame of the molecule to be the one other frames will compare to. The selection contains the atoms in the backbone of the protein, excluding hydrogens:  
`set frame0 [atomselect top "protein and backbone and noh" frame 0]`

- Text after # denotes comment.  

```
# rmsd calculation loop
```
- Loop over all frames in the trajectory:  

```
for { set i 1 } { $i ≤ $nf } { incr i } {
```
- Make the same selection as before for the current frame (the frame to be compared).  

```
set sel [atomselect top "protein and backbone and noh" frame $i]
```
- Calculate the matrix that will fit both selections. Apply this matrix to the second selection to align the molecules:  

```
$sel move [measure fit $sel $frame0]
```
- Calculate the RMSD value between these two selections, and write it to file:  

```
puts $outfile "[measure rmsd $sel $frame0]"
```

You can use the script for the system to test for equilibration.

- 9 Open the TkCon window of VMD by clicking Extensions → tkcon in the VMD Main window. In the TkCon window, type `source rmsd.tcl`. This will perform all the commands in the script. The script will write a file `rmsd.dat` that will contain the value of the RMSD of the protein backbone against time.

Outside of VMD, you can use some plotting program to see this data. Examples of these are gnuplot, xmgrace, Microsoft Excel, Mathematica.

- 10 Use xmgrace to plot the file `rmsd.dat`. In the Unix Terminal window, type `xmgrace rmsd.dat`. The curve does not reveal very much information because of short runtime and infrequent writing of the dcd file. Figure 8 shows a more typical RMSD plot for an equilibrated system along with the plot you have generated. Can you see the first RMSD curve flattening? This means the system is equilibrated.

### 1.7.2 RMSD for Protein without Last 5 Residues

The last five residues are relatively unstable compared to the rest of the protein. How do you think the RMSD should differ in this case than with the entire protein?

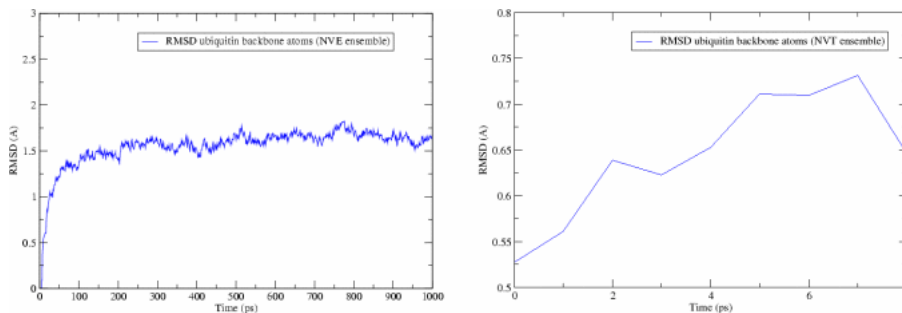


Figure 8: RMSD versus simulation time for a system equilibrating with a longer run time(left) and your system (right)

- 1 You can easily modify the `rmsd.tcl` to plot the RMSD for the protein without the last five residues. Exit `xmgrace` by clicking `File` → `Exit` in the program window. Create an identical file to `rmsd.tcl` with a new name by typing `cp rmsd.tcl rmsd.5.tcl` in the Unix Terminal window.
- 2 Open the new file `rmsd.5.tcl` using `nedit` by typing `nedit rmsd.5.tcl` in the Unix Terminal window.
- 3 Edit the file in the following way:
 

Line 1:	<code>open rmsd.dat w</code>	→	<code>open rmsd.5.dat w</code>
Line 3:	<code>...and noh"</code>	→	<code>...and noh and not (resid 72 to 76)"</code>
Line 6:	<code>...and noh"</code>	→	<code>...and noh and not (resid 72 to 76)"</code>
- 4 Save the file and exit `nedit` by clicking `File` → `Save` and then clicking `File` → `Exit`.
- 5 Type `source rmsd.5.tcl` in the TkCon window of VMD. This will perform all the commands in the script. The script will write a file `rmsd.5.dat` that will contain the value of the RMSD of the protein backbone against time without the last five residues.
- 6 Use `xmgrace` to plot the file `rmsd.5.dat` in the same manner as before with file `rmsd.dat`.
- 7 When you are done analyzing the plot, close `xmgrace`.

## 2 Analysis

This Unit presents a series of examples of how to analyze data produced from NAMD simulations. For these examples you will not run the simulations themselves; instead you will analyze already produced data. Nevertheless, the instructions to set up and run the simulations are included, so you can actually produce the data yourself if time permits.

The steps that must be followed for the analysis are marked with an asterisk (\*). Make sure to go through these steps. We encourage you to at least read the other steps, so you can refer to them later when setting up future simulations.

You will need to use software packages other than NAMD to analyze your data. Specifically, you will use:

- **VMD**, a molecular graphics program. This software is also developed by the Theoretical and Computational Biophysics Group. You can download it for free from [www.ks.uiuc.edu/Research/vmd](http://www.ks.uiuc.edu/Research/vmd). We encourage you to go through the VMD tutorial at [www.ks.uiuc.edu/Research/vmd/tutorial](http://www.ks.uiuc.edu/Research/vmd/tutorial) prior to using the present tutorial.
- a **2-D plotting program** such as xmgrace, gnuplot, Matlab, Mathematica, or Excel. You will find detailed instructions to go through the exercises using xmgrace, but you are encouraged to use your preferred method.
- **other NAMD tools** that you can find at [www.ks.uiuc.edu/Research/namd](http://www.ks.uiuc.edu/Research/namd).

The exercises will have detailed explanations of methods used. We encourage, however, that you be bold and try your own way to do the analysis.

### 2.1 Equilibrium

This section contains exercises that will let you analyze properties of a system in equilibrium. The exercises mostly use pre-equilibrated trajectories from different ensembles to calculate properties of the ubiquitin system.

#### 2.1.1 RMSD for individual residues

**Objective:** Find the average RMSD over time of each residue in the protein using VMD. Display the protein with the residues colored according to this value.

\* 1 Go to the directory `2-1-rmsd/`

You will use a script within VMD that will allow you to compute the average RMSD of each residue in your protein, and assign this value to the B column of the pdb file, which is typically reserved for the temperature factor of each

residue.

In the previous unit, you had VMD open with the equilibration trajectory of the ubiquitin cubic system loaded. You will repeat this process:

- \* **2** Launch VMD
- \* **3** You will now load the structure file. First, open the Molecule File Browser from the File → New Molecule... menu item. Browse for and Load the file `ubq_wb.psf` in the directory `common` use the button `Up one directory` to find it).
- \* **4** The menu at the top of the window should now show `0: ubq_wb.psf`. This ensures that the next file that you load will be added to that molecule (molecule ID 0). Now, browse for the file `ubq_wb_eq.dcd` in the directory `1-3-box/` and click on `Load` again.
- \* **5** Open the TkCon console by choosing the the `Extensions → tkcon` menu item in VMD.
- \* **6** The script you will use is called `residue_rmsd.tcl`. In the TkCon window, type:

```
source residue_rmsd.tcl
```

This command itself does not actually perform any calculations. Instead, it executes the script `residue_rmsd.tcl` which contains a procedure called `rmsd_residue_over_time`. Calling this procedure will calculate the average RMSD for each residue you select over all the frames in a trajectory. The procedure is called as:

```
rmsd_residue_over_time mol sel_resid
```

where *mol* is the molecule in VMD that you select (normally, the top molecule), and *sel\_resid* is a list of the residue numbers in that selection. The procedure employs the equations for RMSD shown in Unit 1.

- \* **7** For this example, you will select all the residues in the protein. The list of residue numbers can be obtained by typing in the TkCon:

```
set sel_resid [[atomselect top "protein and alpha"] get resid]
```

The above command will get the residue numbers of all the  $\alpha$ -carbons in the protein (since there is just one and only one  $\alpha$ -carbon per residue, it is a good option). The command will create a list of residue numbers in the variable `$sel_resid`.
- \* **8** Call the procedure to calculate the RMSD values of all atoms in the newly created selection:



```
rmsd_residue_over_time top $sel_resid
```

You should see the molecule wiggle as each frame gets aligned to the initial structure. At the end of the calculation, you will get a list of the average RMSD per residue. This data is also printed to the file `residue_rmsd.dat`.

The procedure also sets the value of the B column of all the atoms of the residues in the selection to the computed RMSD value. You will now color the protein according to this value. You will be able to recognize which residues are free to move more and which ones move less during the equilibration.

In order to use these new values to learn about the mobility of the residues, you will create a representation of the protein colored by B value.



**The PDB B-factor field.** The “B” field of a PDB file typically stores the “temperature factor” for a crystal structure and is read into VMD’s “Beta” field. Since we are not currently interested in this information, we can use this field to store our own numerical values. VMD has a “Beta” coloring mode, which you will soon use, and which colors atoms according to their B-factors. Thus, by replacing the Beta values for various atoms, you can control the color in which they are drawn. This is very useful when you want to show a property of the system that you have computed.

- \* **9** Open the Representations window using the Graphics → Representations... menu item.
- \* **10** In the Atom Selection window, type `protein`. Choose Tube as Drawing Method, and in the Coloring Method drop-down menu, choose Beta. Now, click on the Trajectory tab, and in the Color Scale Data Range, type 0.40 and 1.00. Click the Set button.

You should now see the protein colored according to average RMSD values. The residues displayed in blue are more mobile while the ones in red move less. This is counter intuitive, so you will change the color scale.

- \* **11** Chose the Graphics → Colors... menu item. Choose the Color Scale tab. In the Method pull-down menu, choose BGR. Your figure should now look similar to Fig. 9.
- \* **12** Look at the RMSD value per residue by typing in a Unix Terminal window (make sure you are in the directory `2-1-rmsd/`):

```
xmgrace residue_rmsd.dat
```

Take a look at the RMSD distribution. You can see regions where a set of residues shows less mobility. Compare with the structural feature of your protein. You will find a correlation between the location of these regions and secondary structure features like  $\alpha$  helices or  $\beta$  sheets.

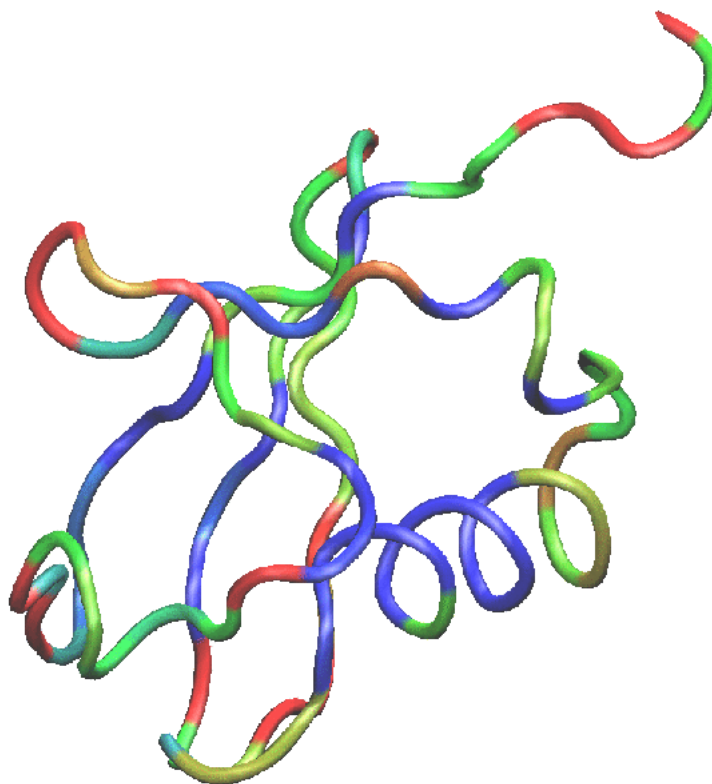


Figure 9: Ubiquitin colored by the average RMSD per residue. Red denotes more mobile residues, blue residues who moved less during equilibration.

### 2.1.2 Maxwell-Boltzmann Energy Distribution

**Objective:** Confirm that the kinetic energy distribution of the atoms in a system corresponds to the Maxwell distribution for a given temperature.

\* 1 Change to the directory 2-2-maxwell/

For this exercise, you will use the last step of the equilibration of ubiquitin. Specifically, you need the file that contains the velocities of every atom in the last frame of this equilibration. The name of the file is `ubq_wb_eq.restart.vel`, you created it in directory 1-3-box/. We will get the necessary information from

this file and the structure file `ubq_wb.psf` with the help of VMD.

- 2 Launch VMD.
- 3 You will now load the structure file. First, open the Molecule File Browser from the `File → New Molecule...` menu item. Browse for and Load the file `ubq_wb.psf` in the directory `common` (use the button `Up` one directory to find it).
- 4 The menu at the top of the window should now show `0: ubq_wb.psf`. This ensures that the next file that you load will be added to that molecule (molecule ID 0). Now, browse for the file `ubq_wb_eq.restart.vel` in the directory `1-3-box/`. This time you need to select the type of file. In the `Determine file type:` pull-down menu, choose `namdbin`, and click on `Load` again.

The molecule you loaded looks terrible! That is because VMD is reading the velocities as if they were coordinates. It does not matter how it looks, because what we want is to make VMD write a useful file: one that contains the masses and velocities for every atom.

- 5 To execute tcl commands, you will use VMD's TkCon window. Open it by selecting the `Extensions → tkcon` menu item. A console window, the TkCon window, should appear with a prompt. You can now start entering Tcl/Tk commands into it.
- 6 First, you will create a selection with all the atoms in the system. Do this by typing in the TkCon window:

```
set all [atomselect top all]
```

- 7 Now, you will open a file `energy.dat` for writing:

```
set fil [open energy.dat w]
```

- 8 For each atom in the system, calculate the kinetic energy  $\epsilon_k = \frac{1}{2}mv^2$ , and write it to a file

```
foreach m [$all get mass] v [$all get {x y z}] {  
    puts $fil [expr 0.5 * $m * [vecdot $v $v] ]  
}
```

- 9 Close the file:

```
close $fil
```

- 10** Outside VMD, take a look at your new `energy.dat` file. You can do this by typing in a Unix Terminal window (make sure you are in the proper directory, i.e. use the same Unix Terminal window from where you launched VMD):

```
nedit energy.dat
```

It should look something like:

```
0.39707419313
0.391385065921
...
```

- 11** Quit `nedit` and VMD.

- \* **12** If you chose not to go through the steps above, there is a VMD script that reproduces them and will create the file `energy.dat`. In a Unix Terminal Window, type:

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

This will run VMD in text mode, source the script `get_energy.tcl`, and produce the `energy.dat` file.

You now have a file of raw data that you can use to fit the obtained energy distribution to the Maxwell-Boltzmann distribution. You can obtain the temperature of the distribution from that fit. In the rest of this section, we will show how to do this with the 2-D graphics package `xmgrace`. If you are more familiar with another tool, you may use it instead.

- \* **13** In a Unix Terminal window, type `xmgrace`.
- \* **14** Once you see the `xmgrace` window, choose the `Data` → `Import` → `ASCII` menu item. Select the file `energy.dat`. Click on the OK button. You should see a black trace. This is the raw data. You can now close the `Read sets` window.
- \* **15** To look at the distribution of points, you will make a histogram of this data. Choose the `Data` → `Transformations` → `Histograms...` menu item. In the `Source` → `Set` window, click on the first line, in order to make a histogram of the data you just loaded.
- \* **16** Click on the `Normalize` option, and fill in `Start: 0`, `Stop at: 10`, and `# of bins: 100`. Click `Apply`.
- \* **17** You have created a plot, but you cannot see it yet. Use the right button of your mouse to click on the first set (in the `Source` → `Set` window). Click on `Hide`. Now, go to the `Main` window and click on the button labeled `AS`, that will resize the plot to fit the existing values. This is your distribution of energies.

## Maxwell-Boltzmann Energy Distribution

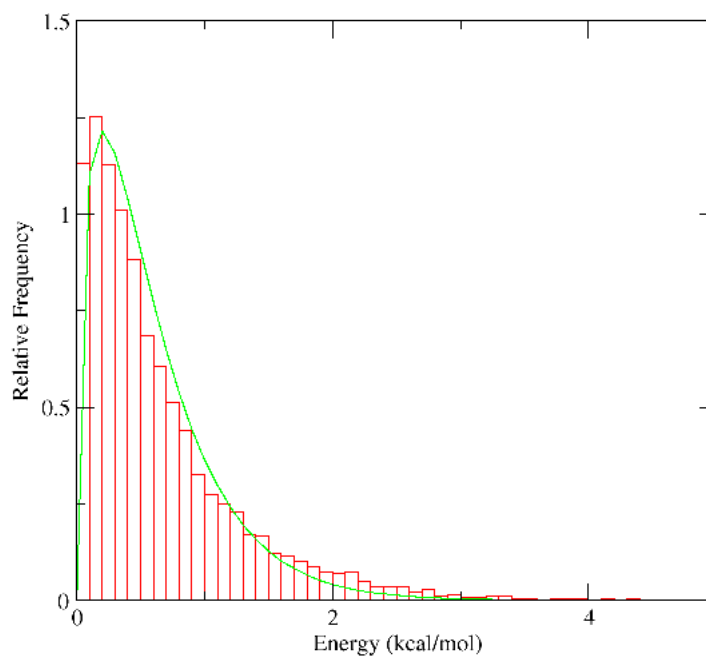
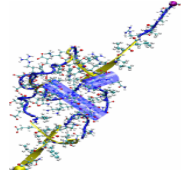


Figure 10: Maxwell-Boltzmann distribution for kinetic energies.

Now, you will fit a Maxwell-Boltzmann distribution to this plot and find out the temperature that the distribution corresponds to. (Hopefully, the temperature you wanted to have in your simulation!)



**Maxwell-Boltzmann distribution for kinetic energy.**

$$f(\epsilon_k) = \frac{2}{\sqrt{\pi}} \frac{1}{(k_B T)^{\frac{3}{2}}} \sqrt{\epsilon_k} \exp\left(-\frac{\epsilon_k}{k_B T}\right) \quad (1)$$

\* 18 Choose the Data → Transformations → Non-linear curve fitting... menu item. In the Source → Set window, click on the last line, which corresponds to the histogram you created.

\* 19 In the Main tab, type in the Formula window:

$$y = (2 / \sqrt{\pi * a0^3}) * \sqrt{x} * \exp(-x / a0)$$

This will fit the curve and get a fit for the parameter  $a0$ , which corresponds to  $k_B T$  in units of kcal/mol. (These are the energy units that NAMD uses)

- \* **20** In the **Parameters** drop-down menu, choose 1. Fill-in forms will appear below. You can give an initial value to  $A0$ , so that the iterations will look for a value in the vicinity of your initial guess. In the  $A0$  window, type 0.6, which in kcal/mol corresponds to a temperature of  $\sim 300K$ .
- \* **21** Click on the **Apply** button. This will open a window with the new value for  $a0$ , as well as some statistical measures, including the **Correlation Coefficient**, which is a measure of the fit (better fit as it approaches 1). You can click on the **Apply** button of the **Non-linear fit** window several times to obtain a better fit.
- \* **22** The value of  $a0$  you obtained corresponds to  $k_B T$ . Obtain the temperature  $T$  for this distribution with  $k_B = 0.00198657 \frac{kcal}{molK}$ . Your result is hopefully close to 310K!

The deviation from your value and the expected one is due to the poor sampling as well as to the finite size of the protein; the latter implies that temperature can be “measured” only with an accuracy of about  $\pm 10K$  even in case of perfect sampling.

### 2.1.3 Energies

**Objective:** Plot the various energies (kinetic and the different internal energies) as a function of temperature.

You will look at the kinetic energy and all internal energies: bonded (bonds, angles and dihedrals) and non-bonded (electrostatic, van der Waals). You should remember that molecular dynamics treats bonds, angles, dihedrals and impropers as harmonic oscillators. You will determine the values of these energies at different temperatures, and find their dependence on this parameter.

To get values over a range of temperatures, you need to look at the output of several simulations, with the same conditions but different temperature. You were provided with a sample configuration file **energies.conf** for temperature  $T = 300K$ . The content of the file is the same as the equilibration configuration file you ran before, except that this time it will have a different value for the temperature. You were assigned a temperature according to your last name.

- \* **1** Go to the directory **2-3-energies/**
- \* **2** Using a web browser, go to [www.ks.uiuc.edu/~villa/energies.html](http://www.ks.uiuc.edu/~villa/energies.html) and find out which temperature you should simulate.

You need to change the temperature in the script for the temperature you were asked to simulate.

- \* **3** Open a text editor by typing in a Unix Terminal window:

```
edit energies.conf
```

- \* **4** Change the line of the script that contains the value for the temperature by putting your value in there:

```
set temperature          your_temperature
```



**Running.** Run your simulation using the configuration file `energies.conf` and the label `energies.log` for your output log file. See handout for further instructions.



**Long job.** This is a long job. You may start the next exercises now and come back later.



**Trajectory file.** The configuration file you will use is set to not write a `dcd` file, since you will not need this for the analysis you will perform. Make sure to revise this configuration file for later use, since you will very likely need a trajectory file.

When your job simulation is completed, you will need to get the average value of the energies at  $T = \textit{your\_temperature}$ . As you saw, the output file from NAMD has a lot of information. A script called `namdstats` does all the work for you! The usage of `namdstats` is:

```
namdstats [from first][to last] file
```

which calculates the average for all output variables in *file* from frame *first* to frame *last*.

- \* **5** Run `namdstats` for your output file:

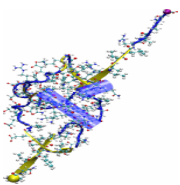
```
namdstats from 101 energies.log
```

Running from frame 101 will avoid the initial energy and the minimization entries (100).

- \* 6 Go again to [www.ks.uiuc.edu/~villa/energies.html](http://www.ks.uiuc.edu/~villa/energies.html) and follow the link for inputting your values. When everyone else is done, we'll show you a plot of the dependence of the energies on temperature. See what you can learn about it from the result.

#### 2.1.4 Temperature distribution

**Objective:** Simulate ubiquitin in an NVE ensemble, and analyze the temperature distribution.



**Temperature.** The temperature of an ensemble is defined as:

$$T = \frac{2}{3Nk_B} \sum_{j=1}^N \frac{1}{2} m_j \bar{v}_j^2 \quad (2)$$

This expression follows from the equipartition theorem of statistical mechanics applied to the kinetic energy

$$\left\langle \sum_{j=1}^N \frac{1}{2} m_j \bar{v}_j^2 \right\rangle = \frac{3}{2} N k_B T \quad (3)$$

where  $N$  is the number of atoms in the system.

- 1 Go to the directory `2-4-temp/`

For this exercise, you need a simulation that is long enough to sample well the quantity of interest, here the fluctuations in kinetic energy, and hence in the temperature. Instead of doing such a simulation yourself, we recommend that you use the provided file `ubq-nve.log`. However, if you choose to run your own simulation, you can find all the necessary files and instructions below. We recommend you to read the whole section even if you don't perform the simulations.

The simulation is performed in the microcanonical ensemble (NVE, i.e., constant  $N$ ,  $V$  and  $E$ ). The configuration file provided to start the NAMD run is called `ubq-nve.conf`. The main features of this configuration file are:

- The simulation takes the restart files from the sphere equilibration simulation performed in Unit 1. The files are located in directory `1-2-sphere/` and are named `ubq_ws`, i.e., `ubq_ws.restart.vel`, etc.
- The initial temperature is determined by the velocity restart file as shown in equation above. this initial temperature corresponds to  $T = 300\text{K}$
- The time step is 2 fs; with `rigidBonds` turned on, this time step is acceptable.
- The simulation will run for 1 ns.





**Running.** Run your simulation using the configuration file `ubq-nve.conf` and the label `ubq-nve.log` for your output log file. See handout for further instructions.

- \* **2** You need to obtain the data for the temperature from the log file. For this you will use a script called `namddat`. Do this by typing:

```
namddat TEMP ubq-nve.log
Your data is now in data.dat.
```

In order to plot the data, you can use `xmgrace`. You need to remove the line with the title as well as the first line of data, that corresponds to the initial temperature  $T=0$ .

- \* **3** Do this using, e.g., `nedit`:

```
nedit data.dat
Remove the first line, save the file as temp.dat and quit nedit. The file
temp.dat can now be read by xmgrace and plotted.
```

- \* **4** Open `xmgrace`.
- \* **5** Choose the `Data` → `Import` → `ASCII` menu item. Select the file `temp.dat`. You should see a black trace. This is a plot of the temperature over time.
- \* **6** To look at the distribution of points, you will make a histogram with this data. Choose the `Data` → `Transformations` → `Histograms...` menu item. In the `Source` → `Set` window, click on the first line, in order to make a histogram of the data you just loaded.
- \* **7** Click on the `Normalize` option, and fill in `Start: 220`, `Stop at: 250`, and `# of bins 100`. Click `Apply`.
- \* **8** You created a plot, but you cannot see it yet. Use the right button of your mouse to click on the first set (in the `Source` → `Set` window). Click on `Hide`. Now, go to the `Main` window and click on the button labeled `AS`, that will resize the plot to fit the existing values. This is your distribution of temperatures.

Does this distribution look any familiar? Your distribution should look like a Gaussian distribution.

**Temperature fluctuations.** The Maxwell distribution for individual kinetic energies  $\epsilon_n$  in its differential form is:

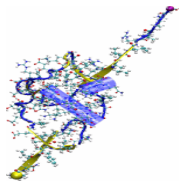
$$dP(\epsilon_n) = \frac{1}{(\pi T_0 \epsilon_n)^{-\frac{1}{2}}} \exp \frac{\epsilon_n}{k_B T_0} d\epsilon_n \quad (4)$$

One can derive:

$$\langle \epsilon_n \rangle = \frac{1}{2} T_0 k_B \quad (5)$$

$$\langle \epsilon_n^2 \rangle = \frac{3}{4} (T_0 k_B)^2 \quad (6)$$

$$\langle \epsilon_n^2 \rangle - \langle \epsilon_n \rangle^2 = \frac{1}{2} (T_0 k_B)^2 \quad (7)$$



The distribution of the total kinetic energy  $E_k = \sum_j \frac{1}{2} m_j v_j^2$ , according to the central limit theorem, is approximately Gaussian

$$P(E_k) = c e^{-\frac{(E_k - \langle E_k \rangle)^2}{3Nk_B^2 T_0^2}} \quad (8)$$

The distribution function for the temperature  $T = 2E_k/3k_B$  fluctuations  $\Delta T = T - T_0$  is then

$$P(\Delta T) = c e^{-\frac{(\Delta T)^2}{2\sigma^2}}, \sigma^2 = 2T^2/3N \quad (9)$$

Note that for  $N \rightarrow \infty$  the distribution is very sharp, but for a finite system it is always broad. For  $T = 300K$  and  $N = 1000$  one has  $\sigma \sim 8K$ .

- \* **9** You will now fit your distribution to a normal distribution. For this, you will use the Non-linear curve fitting feature of `xmgrace`.
- \* **10** Choose the Data  $\rightarrow$  Transformations  $\rightarrow$  Non-linear curve fitting... menu item. In the Source  $\rightarrow$  Set window, click on the last line, which corresponds to the histogram you created.
- \* **11** In the Main tab, type in the Formula window:  
`y= a0 * exp(-(x-a1)^2/a2)`
- \* **12** In the Parameters drop-down menu, choose 3. Fill-in frames will appear below. Give A0 an initial value of 1, A1 an initial value of 235 (you can see that the center of the Gaussian is around that value), and A2 a value of 2. Click several times on the Apply, to get a better fit.

This will fit the curve and get values for the parameters `a0`, `a1` and `a2`, where `a0` is a normalization constant, `a1` the average temperature, and `a2` is  $\sigma^2$ .

The window that appeared contains the new value of the parameters, as well as some statistical measures, including the Correlation Coefficient, which is a measure of the fit.

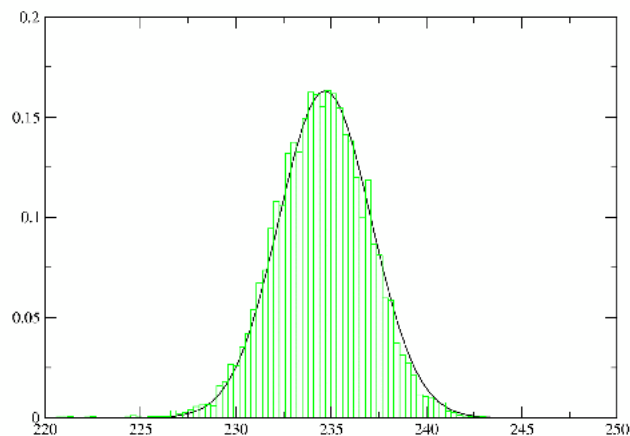


Figure 11: Fluctuation of Temperature

- \* **13** Compare the average value of the temperature obtained by this method with the one you would obtain using `namdstats` by typing in a Unix Terminal window:

```
namdstats ubq-nve.log
```

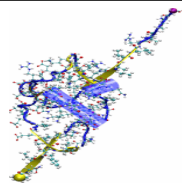
- \* **14** Now, look at the deviation  $\sigma^2 = 2T^2/3N$  which can be calculated from your data. Note how it is dependent on the size of the system!

Take home message: You may note from this example that single proteins, on the one hand, behave like infinite thermodynamic ensembles, reproducing the respective mean temperature; and, on the other hand, they show signs of their finiteness, i.e. temperature fluctuations. In general, it is interesting to note that the velocities of the atoms and, therefore, the kinetic energy of the protein serves as its thermometer!

### 2.1.5 Specific Heat

**Objective:** Find the specific heat of ubiquitin.

Specific heat is an important property of thermodynamic systems. It is the amount of heat needed to raise the temperature of an object per degree of temperature increase per unit mass, and a sensitive measure of the degree of order in a system. (Near a bistable point, the specific heat becomes large.)



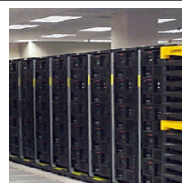
**Energy fluctuations and specific heat.** In the canonical ensemble, the specific heat is defined as:

$$c_V = \frac{\langle E^2 \rangle - \langle E \rangle^2}{k_B T^2} \quad (10)$$

For this exercise, you need to carry out a long simulation in the NVT (canonical) ensemble that samples sufficiently the averages  $\langle E^2 \rangle$  and  $\langle E \rangle$  that arise in the definition of  $c_V$ . To produce the data for the total energy of the molecule over time, you need to work through the steps presented below. This exercise is long and computationally intensive. We strongly recommend that you work only through the steps marked with \*, that correspond to the analysis of the data. Nevertheless, we encourage you to read through the whole method since it will provide you with tools for future analysis.

\* 1 Change to the directory `2-5-spec_heat/`

In order to produce an NVT run, you will restart the equilibration you performed in Unit 1 for the water sphere. All necessary files are present in your directory.



**Running.** Run your simulation using the configuration file `ubq-nvt.conf` and the label `ubq-nvt.log` for your output log file. See handout for further instructions.



**Trajectory writing frequency.** This configuration file will save frames to the `dcd` very often. This will result in large output files. Make sure you revise this configuration file for later use, as normally you won't need this massive kind of output.

You have now an equilibration run of ubiquitin in the NVT ensemble, and you want to look at the fluctuation in the energy of ubiquitin. The problem is that in order to calculate the specific heat for the protein only (as opposed to the whole system), you need the total energies for the atoms in the protein only, and the NAMD output for the run you just performed contains the energies for all atoms in the system.

You would like NAMD to write the energies for only a part of the system. This feature is currently available in NAMD only for interaction energies between

two parts of a system. A way to accomplish this would be to have a system with the protein only, but it does not make sense to equilibrate the protein in this way. However, using a trick, you can make NAMD write the energies for the protein only given the equilibration you already performed. You will first create a dcd with the protein only. For this, you will use a VMD script called `mk_protein_dcd.tcl`. This script writes a pdb for each frame of the trajectory, then loads all the pdb files, and writes a dcd file.

**2** Run this script in the text mode of VMD:

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

Note that this script will take a long time to execute!

Now that you have a dcd file for the protein only, you will use the following trick: for each frame of your pre-equilibrated trajectory, start a NAMD simulation that will run for 0 time steps taking the coordinates from the dcd. This will make NAMD write the energies for every frame on the dcd in the output file.

The configuration file for running NAMD in this setup is `ubq-get-energy.conf`. The way NAMD loops over all frames in the trajectory, taking the coordinates from the current frame and running for 0 time steps is included in this file:

```
# runs 0 time steps for each frame in the dcd
# opens the dcd file to read the coordinates
coorfile open dcd ubq-nvt.dcd
set i 0
while { ![coorfile read] } {
    incr i
    firsttimestep $i
    run 0
}
coorfile close
```



**Running.** Run your simulation using the configuration file `ubq-get-energy.conf` and the label `ubq-get-energy.log` for your output log file. See handout for further instructions.

Now you have a log file with the energy of the protein. You need to retrieve the values for the total energy and put them into a file:

**\* 3** Use `namddat` to get TOTAL:

```
namddat TOTAL ubq-get-energy.log
```

Now that you have a data file `data.dat` with the energy of the protein over time, you can calculate the energy fluctuation. For this you need:

- \* 4 The average energy, which you can get by using the awk script `average.awk` (You can use `namdstats` too) In a Unix Terminal window type:  
`awk -f average.awk data.dat`
- \* 5 The average squared energy, which you can get by using the awk script `squared-average.awk`:  
`awk -f squared-average.awk data.dat`
- \* 6 Calculate the specific heat for ubiquitin using the equation provided, with  $k_B = 0.00198657 \frac{\text{kcal}}{\text{molK}}$  and  $T = 300\text{K}$ . What is your result? Try to convert your results to units of specific heat used everyday, i.e.,  $J/(kg \text{ } ^\circ C)$  using the conversion factor  $1 J = 1.43846 \times 10^{20} \text{ kcal/mol}$ . Note that this units correspond to the specific heat per unit mass. You calculated the fluctuation in the energy of the whole protein, so you must take into account the mass of your protein by dividing your obtained value by  $m = 1.4219 \times 10^{-23} \text{ kg}$ . Compare with some specific heats that are given to you in the table below.

Substance	Specific Heat $J/(kg \text{ } ^\circ C)$
Human Body(average)	3470
Wood	1700
Water	4180
Alcohol	2400
Ice	2100
Gold	130
Strawberries	3890

Table 1: Specific heat of several substances. Data sources: [www.yesican.yorku.ca/home/sh\\_table.html](http://www.yesican.yorku.ca/home/sh_table.html) and [http://www.eng.auburn.edu/users/wfgale/usda\\_course](http://www.eng.auburn.edu/users/wfgale/usda_course)

## 2.2 Non-equilibrium properties of protein

This section contains two exercises that present situations in which the ubiquitin system is not at equilibrium. This section requires some knowledge of the principles of statistics physics, but the results of the simulations can be understood intuitively even without a deep understanding of their principles.

### 2.2.1 Heat Diffusion

**Objective:** Simulate cooling of the ubiquitin molecule and determine, from the results of your simulation, the thermal diffusivity of your system.

For this exercise, you will use the last step of the equilibration of ubiquitin in a water sphere `ubq_ws_eq.restart.coor`. You will use the temperature coupling feature of NAMD to set the temperature of the molecules in the outer layer of the sphere to 200 K. You will determine the thermal diffusivity by monitoring changes of the system's temperature and comparing them to the theoretical expression.

**Heat diffusion.** Time dependence of the local temperature  $T(\vec{r}, t)$  is governed by the heat diffusion equation

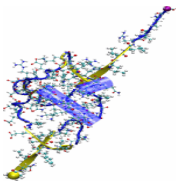
$$\frac{\partial T(\vec{r}, t)}{\partial t} = D\nabla^2 T(\vec{r}, t),$$

subject to the initial and boundary conditions, which for a sphere of radius  $R$  are

$$T(\vec{r}, 0) = \langle T_{sim} \rangle(0) \text{ for } r < R,$$

$$T(R, t) = T_{bath},$$

where  $D$  is the thermal diffusivity,  $T_{sim}$  is the initial temperature of the sphere, and  $T_{bath}$  is the temperature of the sphere's boundary. The average temperature of such a system depends on time as

$$\langle T \rangle(t) = T_{bath} + 6 \frac{T_{sim} - T_{bath}}{\pi^2} \sum_{n=1}^{\infty} \frac{1}{n^2} \exp \left[ - \left( \frac{n\pi}{R} \right)^2 Dt \right].$$


\* 1 Change to this problem's directory:

```
cd 2-6-heat_diff/
```

In order to use the temperature coupling feature of NAMD2, you need to create a file which marks the atoms subject to the temperature coupling. The following lines

```
tCouple on
tCoupleTemp 200
tCoupleFile ubq_shell.pdb
tCoupleCol B
```

in the provided configuration file `ubq_cooling.conf` will enable this feature, and set the atoms coupled to the thermal bath as the ones that have a value of 1.00 in the B column of the `ubq_shell.pdb`

2 Launch VMD.

3 Open the VMD TkCon window by choosing the Extensions → tkcon menu item.

4 Load the system into VMD by typing the following in the VMD tkcon window:

```
mol load psf ubq_ws.psf namdbin ubq_ws_eq.restart.coor
```

5 Select all atoms in the system:

```
set selALL [atomselect top all]
```

6 Find the center of the system:

```
set center [measure center $selALL weight mass]
```

7 Find  $x$ ,  $y$  and  $z$  coordinates of the system's center:

```
foreach {xmass ymass zmass} $center { break }
```

8 Select atoms in the outer layer:

```
set shellSel [atomselect top "not ( sqr(x-$xmass)  
+ sqr(y-$ymass) + sqr(z-$zmass) <= sqr(22) ) "]
```

9 Set beta parameters of the atoms in this selection to 1.00:

```
$shellSel set beta 1.00
```

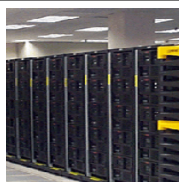
10 Select the entire system again:

```
set selALL [atomselect top all]
```

11 Create the pdb file that marks the atoms in the outer layer by “1.00” in the beta column:

```
$selALL writepdb ubq_shell.pdb
```

12 You can now quit VMD.



**Running.**

Run NAMD for the configuration file `ubq_cooling.conf`. Make sure that the output file is `ubq_cooling.log`.

\* 13 Use `namdplot` to find out how the system's temperature changes with time:

```
namdplot TEMP ubq_cooling.log
```

The `xmgrace` program window will pop up.

\* 14 In the `xmgrace` program, choose the `Data` → `Transformations` → `Geometrical transform ...` menu item. In the `Scale X:` window, type 2.0, which corresponds to the time step used in your simulation. Press `Accept`, which will rescale the plot such that the  $x$ -axis is now in units of fs.



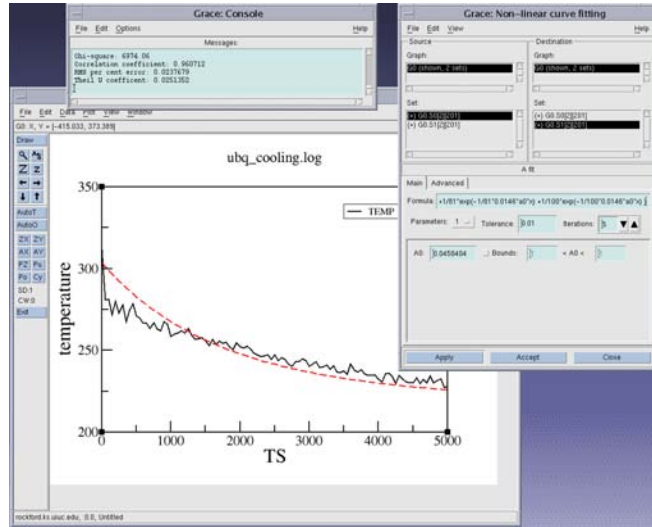


Figure 12: Cooling of ubiquitin in a water sphere. The dashed line shows a non-linear fit of the data to the theoretical expression.

- \* 15 You will now fit the simulated temperature dependence to the theoretical expression (see the Science Box). For this, you will use the **Non-linear curve fitting** feature of **xmgrace**.
- \* 16 Choose the **Data** → **Transformations** → **Non-linear curve fitting...** menu item. In the **Source** → **Set** window, click on the last line, which corresponds to the simulated time dependence.
- \* 17 In the **Main** tab, type or copy and paste over the **Formula** window (make sure it is a one line):
 
$$y = 200 + 66.87 * (\exp(-0.0146 * a_0 * x) + 0.25 * \exp(-0.25 * 0.0146 * a_0 * x) + 1/9 * \exp(-1/9 * 0.0146 * a_0 * x) + 1/16 * \exp(-1/16 * 0.0146 * a_0 * x) + 1/25 * \exp(-1/25 * 0.0146 * a_0 * x) + 1/36 * \exp(-1/36 * 0.0146 * a_0 * x) + 1/49 * \exp(-1/49 * 0.0146 * a_0 * x) + 1/64 * \exp(-1/64 * 0.0146 * a_0 * x) + 1/81 * \exp(-1/81 * 0.0146 * a_0 * x) + 1/100 * \exp(-1/100 * 0.0146 * a_0 * x) )$$
- \* 18 In the **Parameters** drop-down menu, choose 1. Windows will appear below. Click several times on **Apply**, to get a better fit. This will fit the curve and get a value for the parameter **a0**, which is the thermal diffusivity.
- \* 19 Multiply the value of the **a0** parameter by 0.1 to get the thermal diffusivity in  $\text{cm}^2\text{s}^{-1}$ . You should get a value of around  $0.45 \times 10^{-2} \text{cm}^2\text{s}^{-1}$ . How does your result compare to the thermal diffusivity of water  $D = 1.4 \times 10^{-3} \text{cm}^2\text{s}^{-1}$ ?

- 20 Compute the thermal conductivity of the system,  $K$ , by the following formula:  $K = \rho c_V D$ , where  $c_V$  is the specific heat (that was calculated in Section 2.1.5) and  $D$  is the thermal diffusivity (assume that the density of the system,  $\rho$ , is 1 g/mL).

### 2.2.2 Temperature echoes

**Objective:** Demonstrate the coherent dynamics of proteins by generating temperature echoes using MD simulations.

The motions of atoms in globular proteins (e.g., ubiquitin), referred to as internal dynamics, comprise a wide range of time scales, from high frequency vibrations about their equilibrium positions with periods of several femtoseconds to slow collective motions which require seconds or more, leading to deformations of the entire protein. The internal dynamics of these proteins on a picosecond time scale (high frequency) can be described as a collection of weakly interacting harmonic oscillators referred to as normal modes. Since normal modes are formed by linear superposition of a large number of individual atomic oscillations, it is not surprising that the internal dynamics of proteins on this time scale has a delocalized character throughout the protein. The situation is similar to the lattice vibrations (phonons) in a crystalline solid. Experimentally, there exist ways to synchronize, through a suitable signal or perturbation, these normal modes, forcing the system in a so-called (phase) coherent state, in which normal modes oscillate in phase. The degree of coherence of the system can be probed with a second signal which through interference with the coherent normal modes may lead to resonances, referred to as echoes, which can be detected experimentally. However, the coherence of atomic motions in proteins decay through non-linear contributions to forces between atoms. The decay of coherence develops on a time scale  $\tau_d$  which can be probed, e.g., by means of temperature echoes, and can be described by employing MD simulations. In a temperature echo the coherence of the system is probed by reassigning the same atomic velocities the system had at an earlier time and then looking to an echo in the temperature at time  $\tau_e$ , as a result of such reassignment. An example is shown in Fig.13. At time  $t_1 = 0$  the velocities of all atoms in the system are quenched; then, at  $t_2 = \tau$  the atomic velocities are reassigning again (quenched) to their values at time  $t_1 = 0$ . As a result, a temperature echo, i.e., a sharp dip in  $T(t)$ , is detected at a subsequent time  $\tau_e = \tau$  after  $t_2$  (i.e., at time  $t = 2\tau$ ).

In this section, you will employ MD simulations to generate temperature echoes in ubiquitin by applying the velocity reassignment just described. By modeling ubiquitin as a large collection of weakly interacting harmonic oscillators (normal modes), you will find that:

- the dephasing time  $\tau_d$  of the oscillators is about one picosecond;

- the temperature echo can be expressed in terms of the temperature auto-correlation function; and
- the depth  $\Delta T(\tau_e)$  of the echoes decay exponentially with the delay time  $\tau_e$ , i.e.,  $\Delta T(\tau_e) \propto \exp(-\tau_e/\tau_d)$ , the exponential decay being determined by the dephasing time  $\tau_d$ .

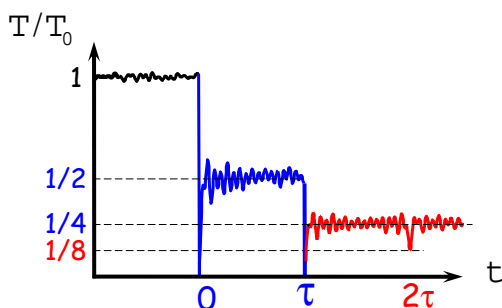
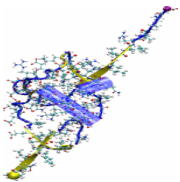


Figure 13: Temperature quench echo.

First you will generate temperature echoes in MD simulations, and then you will analyze the echoes in the framework of the normal mode approximation.



**Temperature echoes.** Temperature echoes are generated by reassigning twice, with a given time delay  $\tau$ , the same (or scaled with the same constant factor) set of velocities to all the atoms of the protein. As a result, in general, the time evolution of the kinetic temperature

$$T(t) = \frac{2}{(3N-6)k_B T} \sum_{i=1}^{3N-6} \frac{m_i v_i^2}{2}, \quad (11)$$

will acquire sharp features at  $t = 0$  and  $t = \tau$ , due to the corresponding energy release or absorption. The spontaneous reappearance of a similar sharp, but less intense, feature in  $T(t)$  at time(s)  $t > \tau$  is referred to as temperature echo(es) (Fig 13).

The phenomenon of temperature echoes has a simple explanation within the normal modes description of a protein.

The first velocity reassignment enforces phase coherence for the oscillators. Because of the frequency dispersion of the normal modes (as well as, the deviation from the harmonic approximation) the phase coherence of the protein will decay in time with a characteristic dephasing time  $\tau_d \sim 1\text{ps}$ .

The second velocity reassignment after a delay time  $\tau$  is a “probing signal” which will test the degree of coherence of the system at the instant of time it

was applied. The depth of the echo and the instant of time at which it occurs are quantitative characteristics of the coherence of the internal dynamics of proteins.

In order to generate temperature echoes, one needs to equilibrate ubiquitin at the desired initial temperature  $T_0 = 300\text{K}$ , e.g., by using the velocity rescaling method. Once the system is equilibrated, the thermostat is removed and all the following simulations are carried out in the microcanonical (NEV) ensemble.

In this problem, you will consider the simplest temperature echo: the set of velocities of the second reassignment is exactly the same as the first one, i.e., without scaling.

### Temperature quench echo

The temperature quench echo is obtained by resetting to zero the velocity of all atoms of the protein at both times  $t_1 = 0$  and  $t_2 = \tau$ . You should start from the pre-equilibrated protein in vacuum at  $T_0 = 300\text{K}$ ; the required files are located in the `common` directory.

You need to run the simulations in the microcanonical ensemble (NVE) by using the configuration file `equil.conf` in the `2-7-echoes/01_equil_NVE` directory. The simulation will run for 500 time steps (fs).

Let's take a look at some features in the configuration file:

- The simulation takes restart files, i.e., the coordinate file and the velocity file, from an equilibration simulation at  $T = 300\text{K}$ .

```
# Continuing a job from the restart files
set inputname ../common/ubi_equil
binCoordinates $inputname.restart.coor
binvelocities $inputname.restart.vel
```

- Notice that the `temperature` line is commented out. The initial temperature is pre-defined by the initial velocities.  
`#temperature $temperature`
- Since you are running an NVE simulation, there is no temperature coupling or pressure control.
- The time step is 1fs, and the `rigidBonds` option is turned off.



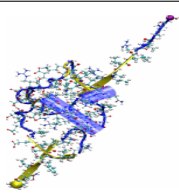
**Running.** Run the simulation with the configuration file `equil.conf` and the label `equil.log` for your output log file. See handout for further instructions.

- 1 Once your job is done, you need to get the temperature data from the output file. Do this by using the script `gettemp`:

```
gettemp equil.log > temp1.dat
```

This will write the data into the file `temp1.dat`.

You will now calculate the temperature autocorrelation function, which will be used later to analyze the temperature echoes.



### Temperature Autocorrelation Function.

$$C_{T,T}(t) = \frac{\langle T(t)T(0) \rangle - \langle T(0) \rangle^2}{\langle [T(t)]^2 \rangle - \langle [T(0)] \rangle^2} \approx \exp(-t/\tau_0) \quad (12)$$

- 2 Start Matlab, by typing in a terminal:

```
tbss> matlab -nosplash &
```

- 3 In the Command Window, type:

```
>> data_analysis
```

This will calculate the temperature autocorrelation function and put it in the file `auto_tmp.dat`.

A plot of the autocorrelation function will appear on your screen (c.f. Fig. 14) in a dotted curve. The solid curve is a fit to an exponential function of Eq. 12. The decay (temperature autocorrelation) time  $\tau_0$  from this exponential is calculated and displayed in the Command Window. This value will be important for a later analysis of your simulations. The value is stored in the file `tau0.dat`.

- 4 Go to the next directory: `02_quencha` by typing in a terminal:

```
cd ../02_quencha
```

- 5 You will make the temperature quenching experiment for a particular  $\tau$  that will be assigned to you. Go to

[http://www.ks.uiuc.edu/~deyulu/thermal\\_echo.html](http://www.ks.uiuc.edu/~deyulu/thermal_echo.html)

to find out which value you should use.

Once you know your value for  $\tau$ , you are going to quench the temperature (set it to zero), and monitor the recovery of the system as it runs for  $\tau$  time steps.

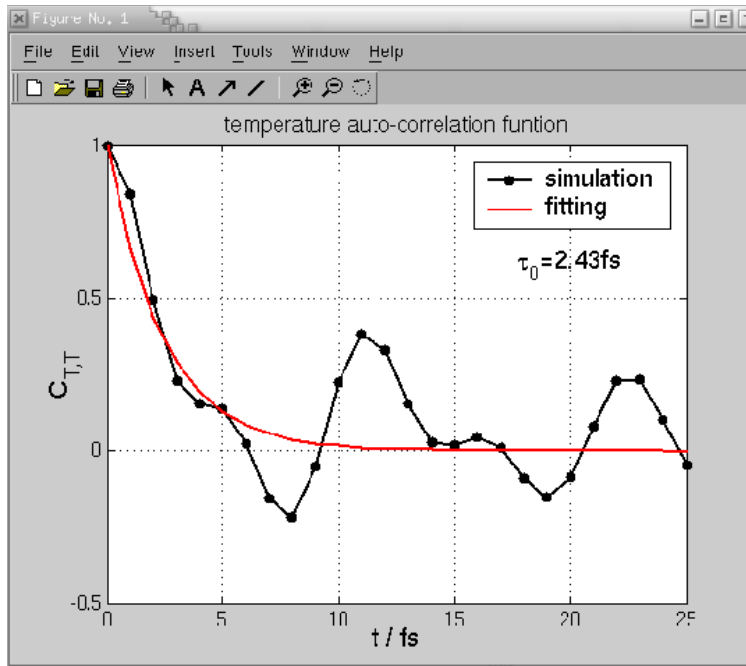


Figure 14: Temperature autocorrelation function

6 Open your configuration file by typing in a terminal:

```
nedit echo.conf
```

7 Change the value of tau to the value you were assigned:

```
set tau 200 ;# here you want to assign your tau value
```

Save and close the file.



**Running.** Run the simulation for the configuration file `echo.conf` and the label `echo.log` for your output log file. See handout for further instructions.

8 Get the temperature data from the output file of the simulation by typing:

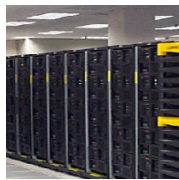
```
gettemp echo.log > temp2.dat
```

This will write the data into the file `temp2.dat`.

Now, you will quench the system for the second time.

9 Change to the directory `03_quenchb/`

- 10 You need to modify the configuration file `echo.conf` with the appropriate value of `tau` again. Note that this time the simulation will run for  $3\tau$  steps:
- ```
run [expr 3*$tau]
```



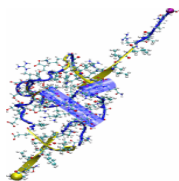
**Running.** Run the simulation for the configuration file `echo.conf` and the label `echo.log` for your output log file. See handout for further instructions.

- 11 Get the temperature data:
- ```
gettemp echo.log > temp3.dat
```
- This will write the data into the file `temp3.dat`.
- 12 Copy the files `temp1.dat` and `temp2.dat` that you generated before to this directory:
- ```
cp ../01_equil_NVE/temp1.dat .
cp ../02_quencha/temp2.dat .
```
- 13 Merge the three temperature data files into `temp.dat`:
- ```
cat temp[1-3].dat > temp.dat
```

You now have the temperature data for the whole temperature quench experiment. You will use Matlab to analyze the data.

- 14 Start Matlab as before by typing:
- ```
matlab -nosplash &
```
- in a terminal.
- 15 Load the file `temp.dat`, by typing in the Matlab Command Window:
- ```
>> load temp.dat
```
- 16 Plot the temperature data by typing:
- ```
>> plot(temp(:,1),temp(:,2))
```
- This command will plot the time in the x axis and the temperature in the y axis (Fig. 15 (A)).

Note anything unusual? You quenched the temperature twice, and yet you see three drops in temperature! The third one is a temperature echo.



**Harmonic Approximation.** The temperature echo phenomenon can be understood by the harmonic approximation of protein motions. In this approximation, the motion of a protein is described by a collection of harmonic oscillators, and the real motion is governed by the statistical average of them. Here we will only give the final expression of the temperature echoes in the harmonic approximation. If we set  $t_1$  as the temporal origin, for time after  $t_2$ , i.e.,  $t > \tau$ , it can be shown that the expression of the temperature echoes from harmonic approximation is

$$T(t) \approx T_0 \left[ \frac{1 + \lambda_1^2 + 2\lambda_2^2}{4} - \frac{1 + \lambda_1^2 - 2\lambda_2^2}{4} C_{T,T}(t - \tau) - \frac{\lambda_1 \lambda_2}{2} C_{T,T}(|t - \frac{3\tau}{2}|) - \frac{1 - \lambda_1^2}{8} C_{T,T}(|t - 2\tau|) \right], \quad (13)$$

where  $\lambda_1 = \sqrt{T_1/T_0}$ , and  $\lambda_2 = \sqrt{T_2/T_0}$ .  $C_{T,T}$  is the temperature auto-correlation function obtained from simulations. For the temperature quench echo  $T_1 = T_2 = 0$ , and (13) simplifies to

$$T(t) \approx \frac{T_0}{4} \left[ 1 - C_{T,T}(t - \tau) - \frac{1}{2} C_{T,T}(|t - 2\tau|) \right]. \quad (14)$$

Next, you will compare  $T(t)$  in the vicinity of the echo ( $t = 2\tau$ ) obtained from the MD simulation with the theoretical prediction (14) involving the temperature autocorrelation function obtained also from the MD simulations. The degree of agreement between these two results is a measure of the accuracy of the harmonic approximation.

- 17** You will use a Matlab script that defines a function called `cmp_echo`. This will plot both the simulated  $T(t)$  and the one given by Eq. 14 (Fig. 15 (B)). Use this function by typing:

```
>> cmp_echo(tau,temp)
```

where  $\tau$  is the delay time `tau` assigned to you. You may also want to use the zoom buttons on the tools menu of the Matlab figure window to magnify the details of the echo.

- 18** You need to find the depth of the echo. Do this by looking at the plot, or by typing the command:

```
>> 75 - min(temp(510+tau:end),2)
```

where 75 is the average temperature after the second quench, and it corresponds to 1/4 of the initial temperature  $T_0=300\text{K}$ . Note that this formula will not catch the echo for larger values of  $\tau$  (e.g.  $\tau=800$  fs).

- 19** Each of you will calculate the echo depth for a particular value of the delay time  $\tau$ . You need to input the echo depth at

[http://www.ks.uiuc.edu/~deyulu/thermal\\_echo.html](http://www.ks.uiuc.edu/~deyulu/thermal_echo.html),



and you will later see a plot showing the data collected from all of you. By fitting them to a single exponential  $\exp(-\tau/\tau_d)$  one obtains the so called *dephasing* time  $\tau_d$ , which is an inherent property of the system.

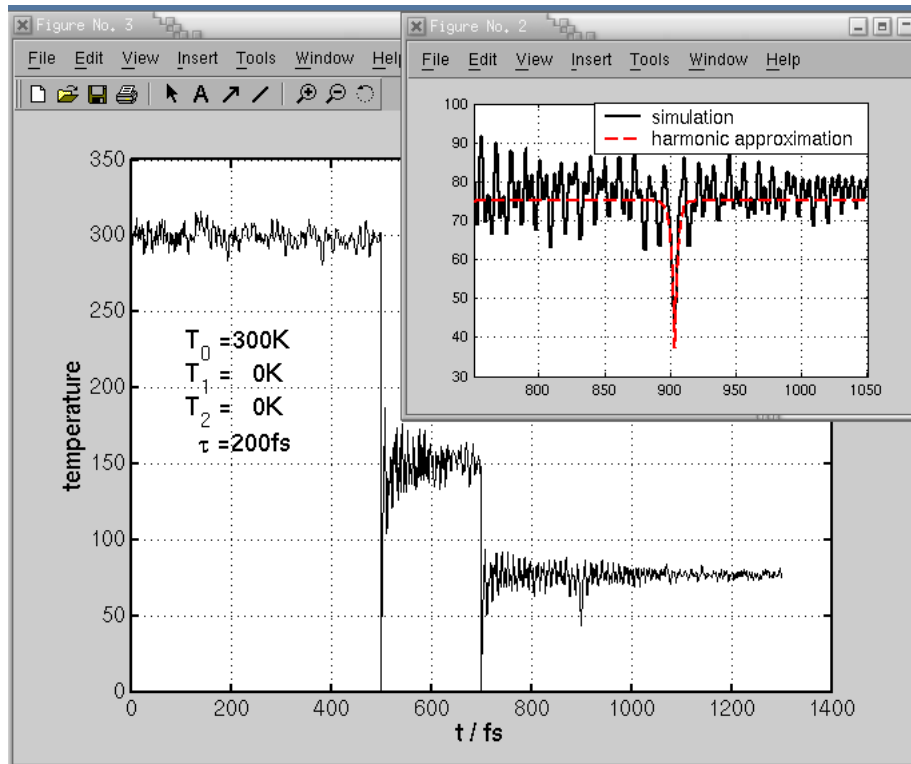


Figure 15: (A) The simulated temperature quench echo. (B) A comparison of the simulation and the expression from harmonic approximation at the vicinity of the echo.

### OPTIONAL EXERCISE: Velocity replacement echo

Here you will generate a temperature echo by replacing the velocities of the atoms at time  $t_2 = \tau$  with the ones you assign to them at time  $t_1$ . Moreover, by choosing these velocities according to the Maxwell-Boltzmann distribution corresponding to  $T_0 = 300\text{K}$ , i.e., the temperature of the equilibrated system ( $t < 0$ ), there will be no discontinuities (jumps) in  $T(t)$  at  $t_1$  and  $t_2$ , yet you will observe a temperature echo at a later time, i.e., a sharp dip in the temperature at time  $t = \tau + \tau_e$ . Your goal is to determine  $\tau_e$  and the depth  $\Delta T$  of the echo. To this end, you will need to follow a similar procedure to the one in the

previous exercise.

**20** Go to the directory `2-7-echoes/04_consta/`.

At  $t_1$  you should reassign the velocities according to a Maxwell-Boltzmann distribution for a temperature  $T_1 = 300K$ . Make sure that you save the re-assigned velocities to a file `300.vel`.

**21** In order to do this, the following is added to the end of your configuration file `echo.conf`:

```
run 0
output 300
```

This settings will not really run a simulation, but will assign initial velocities with the desired distribution and save them to `300.vel`. Following these, the normal `run` command is used to perform a simulation for `tau` time steps:

```
run $tau
```



**Running.** Run the simulation with the configuration file `echo.conf` and the label `echo.log` for your output log file. See handout for further instructions.

**22** Get the temperature data as before and put it in the file `temp2.dat`. Note that at the beginning there is a duplicate entry for the first time step. You should delete one of these:

```
500 300.5656
500 300.5656
501 301.0253
502 302.5395
...
```

**23** Go to the directory `05_constb/`.

**24** For this simulation, you should use the file `300.vel` you generated before as the velocity restart file. This is included in the configuration file `echo.conf` as:

```
velocities ../04_consta/300.vel
```

This will reassign the velocities to the exact same distribution they had at the beginning of the previous simulation. This simulation will run for  $3\tau$  time steps.



**Running.** Run the simulation with configuration file `echo.conf` and the label `echo.log` for your output log file. See handout for further instructions.

- 25 Again get the temperature data and put it in the file `temp3.dat`.
- 26 Copy the files `temp1.dat` and `temp2.dat` into this directory:  

```
cp ../01_equil_NVE/temp1.dat .  
cp ../04_consta/temp2.dat .
```

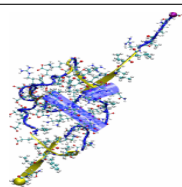
Merge the three temperature data files into `temp.dat`.
- 27 Start Matlab and use the script `cmp_echo.m` to analyze your data.
- 28 Try to locate the position of the echo. This time, the depth of the echo is measured by:  

```
>> 300 - min(temp(510+tau:end,2))
```
- 29 Compare your result with the pre-made results stored in the directory `example_data` for  $\tau = 200$  fs. How would you explain your findings ?

## 3 Steered Molecular Dynamics

*In this section you will learn how to use Steered Molecular Dynamics (SMD). In particular, you will explore elastic properties and unfolding pathways while becoming accustomed to constant velocity and constant force SMD simulations performed by NAMD.*

- The basic idea behind any SMD simulation is to apply an external force to one or more atoms, which we refer to as SMD atoms. In addition, you can keep another group of atoms fixed and study the behaviour of your protein under various conditions.
- In the following simulations, you will fix one atom of ubiquitin and pull on another one and, thereby, stretch and unfold ubiquitin. Before running the actual simulations, you will prepare your system and create the files needed.



**SMD.** Steered Molecular Dynamics allows you to explore biological processes on time scales accessible to molecular dynamics simulations. For instance, unbinding of ligands and conformational changes in biomolecules (like those explored using Atomic Force Microscopy, AFM) can be studied using this technique.

### 3.1 Removing Water Molecules



**Water Molecules.** In actual SMD simulations water plays a fundamental role, e.g., in unfolding pathways for proteins. Since the scope of this tutorial is to introduce NAMD by performing short simulations, we will remove water molecules in the next simulations. However, the reader should be aware that this step is done in order to save time when doing the tutorial. Removing water should be in principle avoided when doing simulations.

- 1 In a Unix Terminal window, set your current directory as `namd-tutorial-files` using the `cd` command. (In this directory you will find the structure already described in Fig. 1.) Open a new session of VMD by typing `vmd`.
- 2 Choose the `File → New Molecule...` menu item of VMD. In the Molecule File Browser use the `Browse...` button to find the file `ubq_ws.psf` in the `common` directory. Load it by pressing the `Load` button.
- 3 In the VMD Main window your molecule should appear (ID 0). Use the mouse to select it; the line will become highlighted in yellow.
- 4 Choose the `File → Load Data Into Molecule...` menu item and, using the `Browse...` and `Load` buttons, load the file `1-2-sphere/ubq_ws_eq.restart.coor`. If you do not find the files, look for the provided output at the subdirectory `1-2-sphere/example-output/`

Now you have the equilibrated ubiquitin in a water sphere loaded in VMD. The next step is to eliminate the water.

- 5 Choose the Extensions → tkcon menu item, and in the VMD TkCon window, type the following commands:

```
set selprotein [atomselect top protein]
$selprotein writepdb common/ubq-ww_eq.pdb
```

You have created a pdb file that contains the equilibrated protein without water.

- 6 Delete the current molecule by choosing the Molecule → Delete Molecule menu item and keep VMD opened.

### 3.2 Constant Velocity Pulling

In this first simulation you will stretch ubiquitin through the constant velocity pulling method.

In this type of simulation the SMD atom is attached to a dummy atom via a virtual spring. This dummy atom is moved at constant velocity and then the force between both is measured using:

$$\vec{F} = -\nabla U \quad (15)$$

$$U = \frac{1}{2}k[v t - (\vec{r} - \vec{r}_0) \cdot \vec{n}]^2 \quad (16)$$

where:

- $U$  → Potential energy.
- $k$  → Spring constant.
- $v$  → Pulling velocity.
- $t$  → Time.
- $\vec{r}$  → Actual position of the SMD atom.
- $\vec{r}_0$  → Initial position of the SMD atom.
- $\vec{n}$  → Direction of pulling.

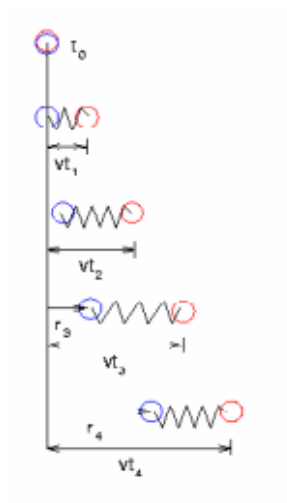


Figure 16: Pulling in a one-dimensional case. The dummy atom is colored red, and the SMD atom blue. As the dummy atom moves at constant velocity the SMD atom experiences a force that depends linearly on the distance between both atoms.



**Pulling more than one atom.** The constant velocity method of pulling can also be used to pull multiple atoms. In that case, the position of the center of mass of the SMD atoms is attached to the dummy atom through the virtual spring.

### 3.2.1 Fixed and SMD Atoms

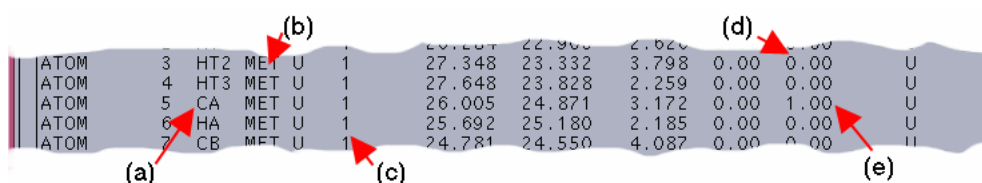
NAMD uses a column of a pdb file to determine which atoms are fixed: all atoms with a value of 1 (or a number different of 0) in a predetermined column will be fixed; atoms with a value of 0 in the same column will not be affected. Here you will use the B column of the pdb file to designate fixed atoms. Now you need to build the respective pdb file using VMD.

- 1 Choose the File → New Molecule... menu item and using the Browse... and Load buttons load the file `common/ubq.psf` that you created in the first unit for ubiquitin in vacuum. (If you did not succeed in generating this file in unit 1, look for `common/example-output/ubq.psf`)
- 2 Select in the VMD Main window the molecule you just loaded and choose the File → Load Data Into Molecule menu item. By using again the Browse... and the Load buttons load the file `common/ubq-ww.eq.pdb`. The equilibrated ubiquitin without water is now available in VMD.
- 3 In the VMD TkCon window, write the following command  
`set allatoms [atomselect top all]`  
This creates a selection called `allatoms` which contains all atoms!
- 4 Type `$allatoms set beta 0`. In this way you set the B column of all atoms equal to 0.
- 5 Type `set fixedatom [atomselect top "resid 1 and name CA"]`. This creates a selection that contains the fixed atom, namely the  $C_{\alpha}$  atom of the first residue.
- 6 Type `$fixedatom set beta 1`. This sets the value of the B column for the fixed atom to 1 and, therefore, NAMD will keep this atom now fixed.

Likewise, NAMD uses another column of a pdb file to set which atom is to be pulled (SMD atom). For this purpose it uses the occupancy column of the pdb file.

- 7 Type `$allatoms set occupancy 0`, so the occupancy column of every atom becomes 0.
- 8 Type `set smdatom [atomselect top "resid 76 and name CA"]`. The `smdatom` selection you created contains the  $C_{\alpha}$  atom of the last residue.

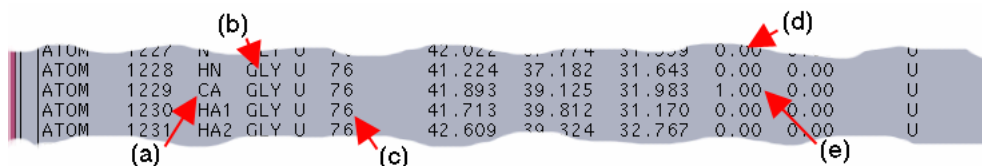
- 9 Type `$smdatom set occupancy 1`. Now, the occupancy column of the `smdatom` selection is 1, and NAMD will pull this atom.
- 10 Finally, type `$allatoms writepdb common/ubq_ww_eq.ref`. This will write a file in `pdb` format that contains all the atoms with the updated B and occupancy columns. Keep VMD opened.
- 11 Using a text editor open the file `ubq_ww_eq.ref` in the `common` directory (for instance type `nedit common/ubq_ww_eq.ref` in the Unix Terminal window) and find the line or row that corresponds to the  $C_\alpha$  atom as shown in Fig. 17 (a) for the methionine (b) residue number 1 (c) or N terminus. In that line you should be able to see how the B column was switched to 1 (e), while the B column for all the other atoms is 0 (d).



|      |   |     |     |   |   |        |        |       |      |      |   |
|------|---|-----|-----|---|---|--------|--------|-------|------|------|---|
| ATOM | 3 | HT2 | MET | U | 1 | 27.348 | 23.332 | 3.798 | 0.00 | 0.00 | U |
| ATOM | 4 | HT3 | MET | U | 1 | 27.648 | 23.828 | 2.259 | 0.00 | 0.00 | U |
| ATOM | 5 | CA  | MET | U | 1 | 26.005 | 24.871 | 3.172 | 0.00 | 1.00 | U |
| ATOM | 6 | HA  | MET | U | 1 | 25.692 | 25.180 | 2.185 | 0.00 | 0.00 | U |
| ATOM | 7 | CB  | MET | U | 1 | 24.781 | 24.550 | 4.087 | 0.00 | 0.00 | U |

Figure 17: Snapshot of the file `ubq_ww_eq.ref` showing the fixed atom row.

- 12 The file you just created (`common/ubq_ww_eq.ref`) also contains the appropriate information required by NAMD to identify the SMD atom. Again, check this by finding the row or line that corresponds to the  $C_\alpha$  atom as shown in Fig. 18 (a) for the glycine (b) residue number 76 (c) or C terminus. This atom, that should have occupancy 1 (e), will be pulled in the simulation.



|      |      |     |     |   |    |        |        |        |      |      |   |
|------|------|-----|-----|---|----|--------|--------|--------|------|------|---|
| ATOM | 1227 | N   | GLY | U | 76 | 42.022 | 37.774 | 31.559 | 0.00 | 0.00 | U |
| ATOM | 1228 | HN  | GLY | U | 76 | 41.224 | 37.182 | 31.643 | 0.00 | 0.00 | U |
| ATOM | 1229 | CA  | GLY | U | 76 | 41.893 | 39.125 | 31.983 | 1.00 | 0.00 | U |
| ATOM | 1230 | HA1 | GLY | U | 76 | 41.713 | 39.812 | 31.170 | 0.00 | 0.00 | U |
| ATOM | 1231 | HA2 | GLY | U | 76 | 42.609 | 39.324 | 32.767 | 0.00 | 0.00 | U |

Figure 18: Snapshot of the file `ubq_ww_eq.ref` showing the SMD atom row.

- 13 When you are done with the comparison of your file with Figs. 17 and 18, close `nedit` or the text editor you have used. Note that coordinates (columns 7, 8, and 9) in your file will not necessarily match the ones shown in the figures because initial velocities in the equilibration were chosen randomly.



**SMD atoms.** Be sure to use the coordinates from the *equilibrated* protein for the file `common/ubq_ww_eq.ref`, so that the initial position of the dummy atom matches the initial coordinates of the SMD atom.



**Fixed atoms.** It is possible to use a column different from the B column to indicate to NAMD the appropriate fixed atoms. In order to do this you have to specify in the configuration file which column you want to use (X, Y, Z, O or B) with the parameter `fixedAtomsCol`.

Now that you have defined the fixed and SMD atom, you need to specify the direction in which the pulling will be performed. This is determined by the direction of the vector that links the fixed and the SMD atoms.

**14** Type the following commands in the VMD TkCon window:

```
set smdpos [lindex [$smdatom get {x y z}] 0]
set fixedpos [lindex [$fixedatom get {x y z}] 0]
vecnorm [vecsub $smdpos $fixedpos]
```

This gives you three numbers that are the  $x$ ,  $y$ , and  $z$ - components of the normalized direction between the fixed and the SMD atom! Keep these  $n_x$ ,  $n_y$  and  $n_z$  saved for the next section.

$$n_x = \quad n_y = \quad n_z =$$

In the provided example (look at Figs. 17 and 18), the result is:

$$(41.893, 39.125, 31.893) - (26.005, 24.871, 3.172) = (15.888, 14.254, 28.811)$$

$$\sqrt{15.888^2 + 14.254^2 + 28.811^2} = 35.856$$

$$n_x = \frac{15.888}{35.856} = 0.443 \quad n_y = \frac{14.254}{35.856} = 0.397 \quad n_z = \frac{28.811}{35.856} = 0.803$$

**15** Delete the current molecule by choosing the `Molecule` → `Delete Molecule` menu item and keep VMD opened.

### 3.2.2 Configuration File

Now you have one of the files required for the intended SMD simulation, namely `common/ubq_ww_eq.ref`. The next step is to create the NAMD configuration file or modify the provided template. Exercise care at this stage and check misspelling of names or commands.



1 Be sure that the current directory in the Unix Terminal window is `namd-tutorial-files`.

2 Copy the file `common/sample.conf` to the directory `3-1-pullcv` and rename it by typing `cp common/sample.conf 3-1-pullcv/ubq_ww_pcv.conf`.

3 Open the configuration file using `nedit` by typing `nedit 3-1-pullcv/ubq_ww_pcv.conf`.

4 As a job description you may write  
`# Constant Velocity Pulling N- C- Termini`

Note that this is just a comment line.

5 In the Adjustable Parameters section you need to change:

```
structure mypsf.psf    → structure ../common/ubq.psf
coordinates mypdb.pdb → coordinates ../common/ubq_ww.eq.pdb
outputName myoutput   → outputName ubq_ww_pcv
```

In this way you are using the equilibrated protein without water in your upcoming simulation. The output files of your simulation will have the prefix `ubq_ww_pcv` in their names.

6 In the Input section change:

```
parameters par_all27_prot_lipid.inp
→ parameters ../common/par_all27_prot_lipid.inp
```

There is no need to modify the Periodic Boundary Conditions, Force Field Parameters, Integrator Parameters or PME sections (The latter is disabled since you are not using periodic boundary conditions).

7 The temperature control should be disabled in order to disturb the movement of the atoms as little as possible. Switch off the Constant Temperature Control by changing:

```
langevin on → langevin off
```

8 Again, there is no need to modify Constant Pressure Control and IMD Settings since they will not be used. However, you have to enable the Fixed Atoms Constraint by changing the following lines:

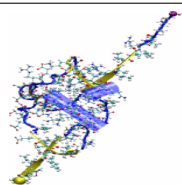
```
if {0} { → if {1} {
fixedAtomsFile myfixedatoms.pdb → fixedAtomsFile
                                   ../common/ubq_ww.eq.ref
```

NAMD will keep fixed the atoms which have a B value of 1 in the file `ubq_ww.eq.ref`.

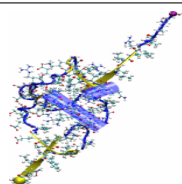
9 In the Extra Parameters section add the following lines:

```
SMD      on
SMDFile  ../common/ubq_ww.eq.ref
SMDk     7
SMDVel   0.002
```

With these configuration file modifications, NAMD will pull atoms which have occupancy 1 in the file `ubq_ww_eq.ref`. The virtual spring between the dummy atom and the SMD atom will have a spring constant of 7 kcal/mol/Å<sup>2</sup>, where 1 kcal/mol = 69.479 pN Å. The pulling will be performed at a constant velocity of 0.002 Å/timestep, equivalent to 1 Å/ps in the present case where the time step is 2 femtoseconds.



**How to choose the spring constant.** The setup of the simulation should be as close as possible to AFM experiments. The spring constant must be high enough that the local unbinding potential is sampled, but must not be too high or the measured force will be dominated by noise.



**Pulling Velocity.** The velocity of pulling is determined basically by the following issues: how close do you want to be to experiment; what total pulling distance and how much computer power do you have available. Therefore, you should choose the smallest velocity that allows you to perform long enough simulations to see the unfolding pathway.

- 10** In the Extra Parameters section you need to write also the direction of pulling by adding the following line:

```
SMDDir  nx ny nz
```

Where  $n_x$ ,  $n_y$  and  $n_z$  should be replaced by the coordinates you calculated at the end of section 3.2.1. (In our example, the direction is  $n_x = 0.443$ ,  $n_y = 0.397$ ,  $n_z = 0.803$ .)

- 11** In the same section, set the frequency in time steps with which the current SMD data values will be printed out by typing:

```
SMDOutputFreq 10
```

- 12** Finally, in the Execution Script section of your configuration file be sure that the minimization is disabled and change the number of time steps of your simulation by replacing:

```
run 50000 → run 30000
```

This is equivalent to 60 ps (and the simulation should not take more than 20 minutes).

- 13** Now your configuration file is ready. SAVE IT and close your text editor.



**Actual Simulations.** As pointed out before, parameters of these simulations are selected so you can run a short SMD simulation. Actual scientific simulations are run for several nanoseconds using pulling velocities of the order of 0.1 Å/ps or smaller in systems that include water molecules (more than 10000 atoms).

### 3.2.3 Running the First SMD Simulation

All the files you need to launch your simulation are now ready. You should have a file called `ubq_ww_pcv.conf` in the `3-1-pullcv` directory and files:

- `ubq.psf`
- `ubq_ww_eq.pdb`
- `ubq_ww_eq.ref`
- `par_all27_prot_lipid.inp`

in the `common` directory. In case that you have not generated these files you can use prepared files available in the directories `3-2-pullcf/example-output` and `common/example-output`.

- 1 Check that you actually have this files in the respective directories.



**Running.** Run your simulation using the configuration file `ubq_ww_pcv.conf` and the label `ubq_ww_pcv.log` for your output log file. See handout for further instructions.

While your simulation is running, some files will be created (as you already learned in Unit 1). The only difference in this case is that in the output file (`ubq_ww_pcv.log`) you will find specific information about the SMD atom and the applied force.

- The additional output line starts with the word SMD in the first column as shown in Fig. 19 (a).
- The second column (b) is the number of timesteps.
- The third, fourth, and fifth columns (c) are the coordinates of the SMD atom.
- The last three columns (d) are the components of the current force applied to the SMD atom in pN.


Since the simulation will take some time, the analysis of the complete output will be explained in section 3.4. Thus, while you wait for the result you can set up your Constant Force simulation.

```

SMD 370 41.065 38.6454 32.951 32.2709 28.952 58.5195
SMD 380 41.0962 38.5292 33.1032 15.0473 13.4997 27.2864
SMD 390 41.1681 38.495 33.3074 -22.1029 -19.8297 -40.081
SMD 400 41.1983 38.428 33.3553 -25.3756 -22.7658 -46.0156
PRESSURE: 400 0 0 0 0 0 0 0 0
GPRESSURE: 400 0 0 0 0 0 0 0 0
ENERGY: 400 234.3106 821.9371 489.8659 64.1836 -2345.2403 -16\
2.4893 0.0000 0.0000 1202.6713 305.2390 395.4340
SMD 410 41.2355 38.456 33.5373 -60.6924 -54.4505 -110.058
SMD 420 41.1923 38.4524 33.6517 -73.9062 -66.3053 -134.02
SMD 430 41.1984 38.4073 33.6689 -71.4504 -64.1021 -129.567
SMD 440 41.1563 38.372 33.7528 -76.7696 -68.8742 -139.213
SMD 450 41.0483 38.3112 33.7452 -58.0002 -52.116 -105.34

```

Figure 19: Typical output of a SMD simulation



**Restarting an SMD simulation.** The position of the dummy atom depends on the velocity of the pulling and the current time. Therefore, if you want to restart an SMD simulation, make sure that you set the `firsttimestep` parameter as the time step of the saved previous configuration.

### 3.3 Constant Force Pulling

Now, you will carry out an SMD simulation that applies a constant force. In this case, the  $C_{\alpha}$  atom of the first residue is again kept fixed, but the SMD atom  $C_{\alpha}$  of the last residue now experiences a constant force in the direction defined by the vector that links both atoms, the fixed and the pulled one. Note that in this case there is no dummy atom or virtual spring.

#### 3.3.1 The SMD Atom

Again, NAMD uses a column of a pdb file to determine which atoms will be fixed and which atoms will be pulled. In addition, another three columns are used to specify the direction of the constant force that will be applied to the SMD atom. Your first task is to build this file.

- 1 In your opened session of VMD, choose the File → New Molecule... menu item and using the Browse... and the Load buttons load the file `common/ubq.psf` located in the `common` directory.
- 2 Using the mouse select the molecule in the VMD Main window and then choose the Load Data Into Molecule... menu item. Again, using the Browse... and the Load buttons load the file `ubq.wv.eq.pdb` located in the `common` directory.

- 3 As you did in the previous section, define your fixed atom by typing the following commands in the VMD TkCon window:

```
set allatoms [atomselect top all]
$allatoms set beta 0
set fixedatom [atomselect top "resid 1 and name CA"]
$fixedatom set beta 1
$allatoms set occupancy 0
```

- 4 The occupancy of the SMD atom will contain the force applied to it. Type

```
set smdatom [atomselect top "resid 76 and name CA"]
$smdatom set occupancy 11.54
```

You have now set the force to 11.54 kcal/mol/Å by entering the value into the occupancy field. It is equivalent to 800 pN.

- 5 The direction of the force will be specified in the coordinates of the SMD atom. Therefore, you have to write the normal vector in the following way:

```
$smdatom set x  $n_x$ 
$smdatom set y  $n_y$ 
$smdatom set z  $n_z$ 
```

where  $n_x$ ,  $n_y$ , and  $n_z$  have to be replaced by the appropriate values you already calculated above (in our example 0.443, 0.397, and 0.803). Since the VMD OpenGL Display interpretes the numbers just entered as coordinates of the SMD atom, it displays now a inaccurate protein structure. This is OK and we apologize for the poor appearance.

- 6 Now, save the file by typing `$allatoms writepdb common/ubq_ww_eq2.ref`
- 7 Delete the current molecule by choosing the Molecule → Delete Molecule menu item and keep VMD opened.

### 3.3.2 Configuration File

As you did before, in the next step you will modify the configuration file in order to set up your constant force simulation.

- 1 Be sure that in the Unix Terminal window your current directory is `namd-tutorial-files`.
- 2 Copy the file `common/sample.conf` to the directory `3-2-pullcf` and rename it by typing `cp common/sample.conf 3-2-pullcf/ubq_ww_pcf.conf`.
- 3 Now, open the configuration file using `nedit` (type `nedit 3-2-pullcf/ubq_ww_pcf.conf`).

4 As a job description you can write

```
# N-C-Termini Constant Force Pulling
```

5 In the Adjustable Parameters section you need to change:

```
structure mypsf.psf    → structure ../common/ubq.psf
coordinates mypdb.pdb  → coordinates ../common/ubq_ww_eq.pdb
outputName myoutput    → outputName ubq_ww_pcf
```

In this way you are using the equilibrated protein without water in your simulation. The output files of your simulation will have the prefix `ubq_ww_pcf` in their names.

6 In the Input section change:

```
parameters par_all27_prot_lipid.inp
→ parameters ../common/par_all27_prot_lipid.inp
```

7 As before, switch off the Constant Temperature Control by changing:

```
langevin on  → langevin off
```

8 Enable the Fixed Atoms Constraint by changing the following lines

```
if {0} {          → if {1} {
fixedAtomsFile myfixedatoms.pdb → fixedAtomsFile
                                   ../common/ubq_ww_eq2.ref
```

NAMD will keep fixed the atoms which have a B value of 1 in the file `../common/ubq_ww_eq2.ref`.

9 In the Extra Parameters section add the following lines:

```
constantforce yes
consforcefile ../common/ubq_ww_eq2.ref
```

NAMD will apply a constant force to the atoms that have occupancy different from 0. The force is calculated from the file as  $(x, y, z) \times O$ , where  $O$  is the value of the occupancy column.

10 Finally, in the Execution Script section of your configuration file be sure that the minimization is disabled and change the number of time steps your simulation will run by replacing:

```
run 50000  → run 30000
```

This is equivalent to 60 ps.

11 Your second configuration file is done. **SAVE IT** and close the text editor.

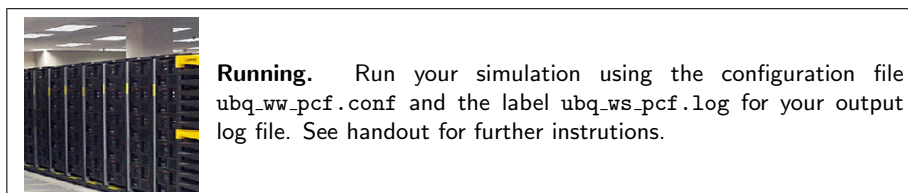
### 3.3.3 Running the Second SMD Simulation

Now, all the files you need to launch your second simulation are ready. You should have a file called `ubq_ww_pcf.conf` in the `3-2-pullcf` directory and files:

- `ubq.psf`

- `ubq_ww_eq.pdb`
- `ubq_ww_eq2.ref`
- `par_all27_prot_lipid.inp`

in the `common` directory. In case that you have not generated these files you can use prepared files available at directories `3-2-pullcf/example-output` and `common/example-output`.



Note that the output file has no extra information in this case, since the applied force is now constant.

## 3.4 Analysis of Results

### 3.4.1 Force Analysis for Constant Velocity Pulling

Hopefully your constant velocity SMD simulation has finished successfully and you can proceed with analysing the generated data, specifically the trajectory and the force applied to the SMD atom. In case that you have not succeeded with your simulations we provide the respective files in the subdirectories `example-output`. For instance, if you do not find a file in the directory `common`, you should find it in the directory `common/example-output`.

- 1 In VMD choose the `New Molecule...` menu item. Using the `Browse...` and the `Load` buttons load the file `ubq.psf` in the `common` directory.
- 2 Now, select with the mouse the molecule (ID 0) in the VMD Main window. Then choose the `Load Data Into Molecule...` menu item and using the `Browse...` and the `Load` buttons load the trajectory file `3-1-pullcv/ubq_ww_pcv.dcd`

In the screen you should be able to see the trajectory of the molecular dynamics simulation. By choosing an appropriate representation (e.g. `Cartoon`) look how the  $\beta$  strands behave along the trajectory and how one end of the protein remains fixed while the other one is pulled as you had set it up. (Fig. 20)

After inspecting your trajectory you should extract the force applied to the SMD atom from the NAMD output file:

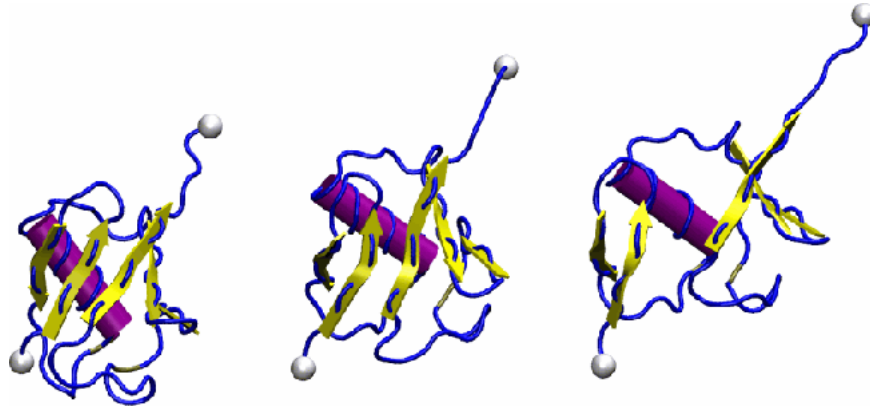


Figure 20: Snapshots of ubiquitin pulling with constant velocity at three different time steps.

- 3 Set your current directory to 3-1-pullcv (type `cd 3-1-pullcv`).
- 4 Type the following command: `cat ubq_ww_pcv.log | awk '{if ($1=="SMD") print $0}'> analysis/smd.dat`  
This creates a file `smd.dat` (in the `analysis` directory) which contains just the SMD information.

In order to obtain the force in the direction of pulling you need to calculate  $\vec{f} \cdot \vec{n}$ , where  $\vec{n}$  is the normalized direction of pulling (in our example it was 0.443, 0.398, 0.803).

- 5 Change your current directory to `analysis` (`cd analysis`) and type the following command:  
`cat smd.dat | awk '{print $1 " " $6*n_x + $7*n_y + $8*n_z}'> ft.dat`,  
where  $n_x$ ,  $n_y$  and  $n_z$  should be replaced by the values you obtained earlier. You have created a file (`ft.dat`) that contains force versus time data. (Note that columns 6, 7, and 8 of the file `smd.dat` are the components of the force)
- 6 Type `xmgrace ft.dat`. A plot of force versus time should appear on your screen (Fig. 21).
- 7 In VMD, look again at the trajectory at different times and try to identify the events that are related to peaks in the force. When you are done close `xmgrace`.



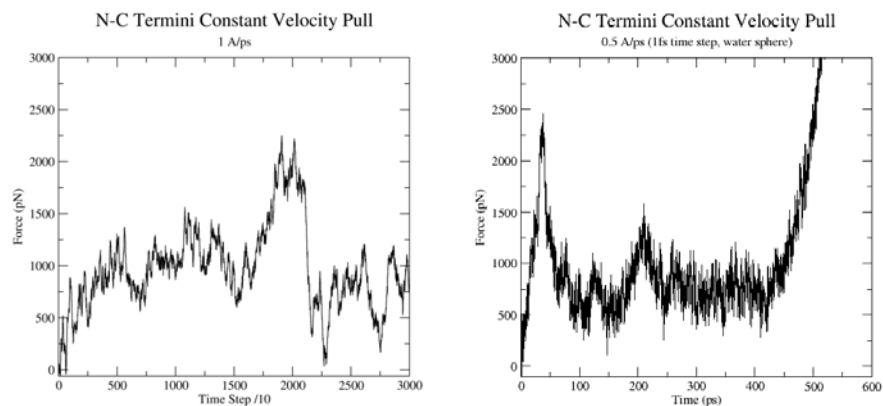
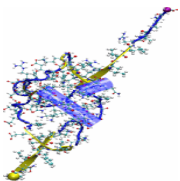


Figure 21: Force vs Time. Your plot should resemble the graph on the left. The graph on the right was obtained in a simulation where ubiquitin was placed in a water sphere ( $\sim 26000$  atoms) and pulled at  $0.5 \text{ \AA/ps}$  using a time step of 1 femtosecond. The results of this last simulation are in better agreement with actual AFM experiments. The force at the end ( $t > 400\text{ps}$ ) increases when the protein becomes completely unfolded. The first peak in the force is associated with the breaking of hydrogen bonds between two  $\beta$  strands.



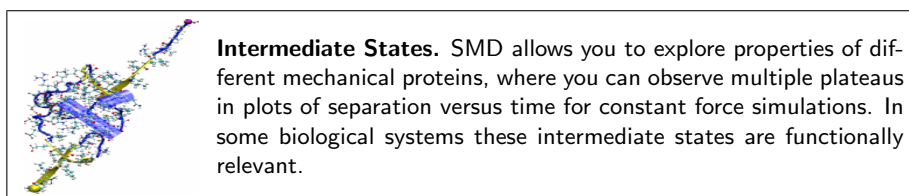
**What can we learn from this graph.** Intermediates states in the unfolding pathway of the protein can be identified using this kind of simulation. A particular useful way to see this is by plotting force vs extension. In addition, microscopic details missed by experiments, like the behaviour of hydrogen bonds and the role of water in the unfolding pathway, can be studied and explored along with the micro-mechanical and elastic properties of the protein.

### 3.4.2 Distance Analysis for Constant Force Pulling

- 1 In VMD, delete the current molecule by choosing the Molecule  $\rightarrow$  Delete Molecule menu item.
- 2 Choose the New Molecule... menu item. Using the Browse... and the Load buttons load the file `ubq.psf` located in the `common` directory.
- 3 Now, select with the mouse the molecule you loaded in the VMD Main window. Then choose the Load Data Into Molecule... menu item and using the Browse... and the Load buttons load the trajectory file `3-2-pullcf/ubq_ww_pcf.dcd`. As mentioned before, in case that you have not accomplished the simulation to generate the file, we provide such file at `3-2-pullcf/example-output/ubq_ww_pcf.dcd`.

In the OpenGL Display window you will see the trajectory of the constant force pulling simulation.

- 4 Once the complete trajectory is loaded, use the slider in the VMD Main window to go back to the first frame.
- 5 Choose the Graphics → Representations... menu item. In the Graphical Representations window choose the Cartoon drawing method.
- 6 Press the Create Rep button in order to create a new representation. In the Selected Atoms text entry delete the word **all** and type **protein** and **resid 1 76** and name **CA**.
- 7 For this representation choose the VDW drawing method. You should be able to see the two  $C_{\alpha}$  atoms at the end of your protein as spheres.
- 8 Hide the first representation by double clicking on it in the Graphical Representations window.
- 9 Click in the OpenGL Display window and press the key 2 (Shortcut to bond labels). Your cursor should look like a cross. Pick the two spheres displayed. A line linking both atoms should appear with the distance between them.
- 10 Choose the Graphics → Labels... menu item. In the Labels window, choose the label type **Bonds**. Select the bond displayed, click on the **Graph** tab and then click on the **Graph** button. This will create a plot of the distance between these two atoms over time (Fig. 22).



This is the end of the NAMD tutorial. We hope you can now make use of VMD and NAMD in your work.

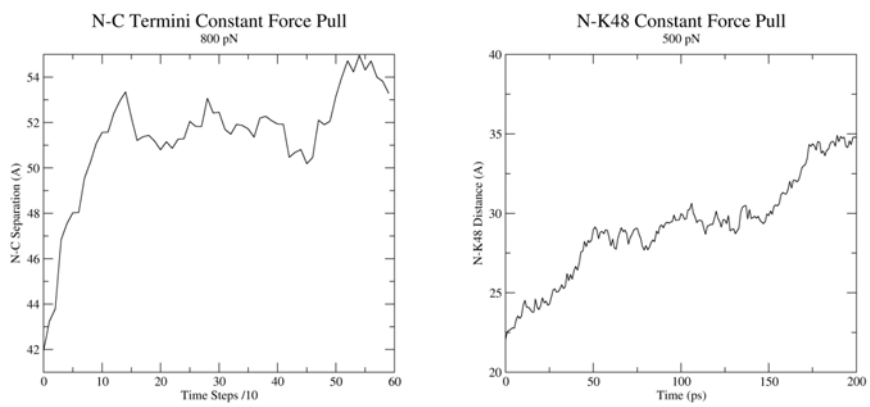


Figure 22: Distance vs Time Steps. Your plot should resemble the graph on the left. The graph on the right was obtained in a simulation where ubiquitin was placed in a water box and pulled from the C-Terminus while keeping the residue K48 fixed. In this case we can observe an intermediate state.

## A PDB Files

The term PDB can refer to the Protein Data Bank (<http://www.rcsb.org/pdb/>), to a data file provided there, or to any file following the PDB format. Files in the PDB include information such as the name of the compound, the species and tissue from which it was obtained, authorship, revision history, journal citation, references, amino acid sequence, stoichiometry, secondary structure locations, crystal lattice and symmetry group, and finally the ATOM and HET-ATOM records containing the coordinates of the protein and any waters, ions, or other heterogeneous atoms in the crystal. Some PDB files include multiple sets of coordinates for some or all atoms. Due to the limits of x-ray crystallography and NMR structure analysis, the coordinates of hydrogen atoms are not included in the PDB.

NAMD and VMD ignore everything in a PDB file except for the ATOM and HETATOM records, and when writing PDB files the ATOM record type is used for all atoms in the system, including solvent and ions. Here are the ATOM records for the first two residues of ubiquitin from the 1UBQ entry in the PDB:

|      |    |     |     |   |        |        |       |      |       |      |    |
|------|----|-----|-----|---|--------|--------|-------|------|-------|------|----|
| ATOM | 1  | N   | MET | 1 | 27.340 | 24.430 | 2.614 | 1.00 | 9.67  | 1UBQ | 71 |
| ATOM | 2  | CA  | MET | 1 | 26.266 | 25.413 | 2.842 | 1.00 | 10.38 | 1UBQ | 72 |
| ATOM | 3  | C   | MET | 1 | 26.913 | 26.639 | 3.531 | 1.00 | 9.62  | 1UBQ | 73 |
| ATOM | 4  | O   | MET | 1 | 27.886 | 26.463 | 4.263 | 1.00 | 9.62  | 1UBQ | 74 |
| ATOM | 5  | CB  | MET | 1 | 25.112 | 24.880 | 3.649 | 1.00 | 13.77 | 1UBQ | 75 |
| ATOM | 6  | CG  | MET | 1 | 25.353 | 24.860 | 5.134 | 1.00 | 16.29 | 1UBQ | 76 |
| ATOM | 7  | SD  | MET | 1 | 23.930 | 23.959 | 5.904 | 1.00 | 17.17 | 1UBQ | 77 |
| ATOM | 8  | CE  | MET | 1 | 24.447 | 23.984 | 7.620 | 1.00 | 16.11 | 1UBQ | 78 |
| ATOM | 9  | N   | GLN | 2 | 26.335 | 27.770 | 3.258 | 1.00 | 9.27  | 1UBQ | 79 |
| ATOM | 10 | CA  | GLN | 2 | 26.850 | 29.021 | 3.898 | 1.00 | 9.07  | 1UBQ | 80 |
| ATOM | 11 | C   | GLN | 2 | 26.100 | 29.253 | 5.202 | 1.00 | 8.72  | 1UBQ | 81 |
| ATOM | 12 | O   | GLN | 2 | 24.865 | 29.024 | 5.330 | 1.00 | 8.22  | 1UBQ | 82 |
| ATOM | 13 | CB  | GLN | 2 | 26.733 | 30.148 | 2.905 | 1.00 | 14.46 | 1UBQ | 83 |
| ATOM | 14 | CG  | GLN | 2 | 26.882 | 31.546 | 3.409 | 1.00 | 17.01 | 1UBQ | 84 |
| ATOM | 15 | CD  | GLN | 2 | 26.786 | 32.562 | 2.270 | 1.00 | 20.10 | 1UBQ | 85 |
| ATOM | 16 | OE1 | GLN | 2 | 27.783 | 33.160 | 1.870 | 1.00 | 21.89 | 1UBQ | 86 |
| ATOM | 17 | NE2 | GLN | 2 | 25.562 | 32.733 | 1.806 | 1.00 | 19.49 | 1UBQ | 87 |

The fields seen here in order from left to right are the record type, atom ID, atom name, residue name, residue ID, x, y, and z coordinates, occupancy, temperature factor (called beta), segment name, and line number.

If this file is loaded into VMD and then written out as a new file, most of the extra information will be removed, the HETATOM records will become ATOM records, and the previously empty chain ID field (between residue name and residue ID) will be set to X (unless present in the original file), and the line number will be omitted, as seen here:

|      |   |    |       |   |        |        |       |      |       |      |  |
|------|---|----|-------|---|--------|--------|-------|------|-------|------|--|
| ATOM | 1 | N  | MET X | 1 | 27.340 | 24.430 | 2.614 | 1.00 | 9.67  | 1UBQ |  |
| ATOM | 2 | CA | MET X | 1 | 26.266 | 25.413 | 2.842 | 1.00 | 10.38 | 1UBQ |  |
| ATOM | 3 | C  | MET X | 1 | 26.913 | 26.639 | 3.531 | 1.00 | 9.62  | 1UBQ |  |

## B PSF Files

A PSF file, also called a protein structure file, contains all of the molecule-specific information needed to apply a particular force field to a molecular system. The CHARMM force field is divided into a topology file, which is needed to generate the PSF file, and a parameter file, which supplies specific numerical values for the generic CHARMM potential function. The topology file defines the atom types used in the force field; the atom names, types, bonds, and partial charges of each residue type; and any patches necessary to link or otherwise mutate these basic residues. The parameter file provides a mapping between bonded and nonbonded interactions involving the various combinations of atom types found in the topology file and specific spring constants and similar parameters for all of the bond, angle, dihedral, improper, and van der Waals terms in the CHARMM potential function.

The PSF file contains five main sections of interest: atoms, bonds, angles, dihedrals, and impropers (dihedral force terms used to maintain planarity). The following is taken from a PSF file for ubiquitin. First is the title and atom records:

PSF

```
1 !NTITLE
REMARKS original generated structure x-plor psf file

1231 !NATOM
1 PROT 1 MET N NH3 -0.300000 14.0070 0
2 PROT 1 MET HT1 HC 0.330000 1.0080 0
3 PROT 1 MET HT2 HC 0.330000 1.0080 0
4 PROT 1 MET HT3 HC 0.330000 1.0080 0
5 PROT 1 MET CA CT1 0.210000 12.0110 0
6 PROT 1 MET HA HB 0.100000 1.0080 0
7 PROT 1 MET CB CT2 -0.180000 12.0110 0
```

The fields in the atom section are atom ID, segment name, residue ID, residue name, atom name, atom type, charge, mass, and an unused 0. PSF files may be in either CHARMM or X-PLOR format, with the CHARMM format using an integer rather than a name for the atom type. NAMD requires the X-PLOR format, which is also more flexible since it is not tied to the specific order of atom types in a single parameter file. NAMD and VMD require that the order of atoms in any PDB, DCD, or other atomic data file exactly matches the order found in the PSF file.

Notice that hydrogen atoms are included, that multiple atoms in a residue may share the same type (e.g., HT1, HT2, and HT3 are of type HC), and that atoms of the same element may have different types (e.g., CA and CB, HA, and HT1). Much of the information in the PDB is also included in the PSF file. Often, the only information from the PDB file used by NAMD is the atomic coordinates. In some cases the contents of the occupancy and beta fields of the PDB file are used as parameters to optional simulation methods.

The covalent bond section lists four pairs of atoms per line:

```
1237 !NBOND: bonds
  1      5      2      1      3      1      4      1
  5      6      7      5      7      8      7      9
 10      7     10     11     10     12     13     10
```

The angle section lists three triples of atoms per line:

```
2257 !NTHETA: angles
  1      5      6      1      5     18      2      1      5
  2      1      4      2      1      3      3      1      5
  3      1      4      4      1      5      5     18     19
```

The dihedral and improper sections list two quadruples of atoms per line:

```
3293 !NPHI: dihedrals
  1      5      7     10      1      5      7      8
  1      5      7      9      1      5     18     20
  1      5     18     19      2      1      5      7

204 !NIMPHI: impropers
 18      5     20     19     20     18     22     21
 30     32     27     31     30     27     32     31
 32     30     33     34     32     30     34     33
```

The order of the atoms within each bond is not significant. The order of atoms within angles, dihedrals, and impropers is significant, but the order can be reversed without affecting the resulting potential. In no case is the relative order of different bonds etc. significant. Since every bond and improper is explicitly listed in the topology file, the atom order within these terms tend to match the original order given. Angles and dihedrals are typically generated automatically and therefore appear in a sorted order.

There is a difference between X-PLOR formatted CHARMM PSF files, which VMD generates with the psfgen module and NAMD uses in conjunction with CHARMM parameter files, and PSF files generated by X-PLOR, which NAMD uses in conjunction with X-PLOR parameter files. Dihedral terms sometimes require multiple sinusoids to represent a torsional potential and therefore multiple parameter sets appear in the parameter file. In PSF files generated by X-PLOR multiple dihedrals would be indicated by duplicate dihedrals in the PSF file. When using CHARMM PSF and parameter files NAMD extracts any multiple dihedral terms directly from the parameter file and each dihedral appears only once in the PSF file.



!MacKerell, Jr., A. D.; Bashford, D.; Bellott, M.; Dunbrack Jr., R.L.;  
!Evanseck, J.D.; Field, M.J.; Fischer, S.; Gao, J.; Guo, H.; Ha, S.;  
!Joseph-McCarthy, D.; Kuchnir, L.; Kuczera, K.; Lau, F.T.K.; Mattos,  
!C.; Michnick, S.; Ngo, T.; Nguyen, D.T.; Prodhom, B.; Reiher, III,  
!W.E.; Roux, B.; Schlenkrich, M.; Smith, J.C.; Stote, R.; Straub, J.;  
!Watanabe, M.; Wiorkiewicz-Kuczera, J.; Yin, D.; Karplus, M. All-atom  
!empirical potential for molecular modeling and dynamics Studies of  
!proteins. Journal of Physical Chemistry B, 1998, 102, 3586-3616.  
!

The topology file must define the type, mass, and charge of every atom in every residue, so that a PSF file can be constructed. While the partial charges assigned to atoms of the same type vary between residues, their masses do not. Therefore, the mass of every atom type is declared once at the beginning of the file in a MASS statement. This statement also pairs an integer with each type name, which is used in CHARMM formatted PSF files, but *not* in the X-PLOR formatted PSF files used by NAMD. The type indices are unique but not necessarily consecutive. Notice in the following except that there are many types of hydrogen and carbon atoms defined, but the atomic masses are the same:

```
MASS    1 H      1.00800 H ! polar H
MASS    2 HC     1.00800 H ! N-ter H
MASS    3 HA     1.00800 H ! nonpolar H
MASS    4 HT     1.00800 H ! TIPS3P WATER HYDROGEN
MASS    5 HP     1.00800 H ! aromatic H
MASS    6 HB     1.00800 H ! backbone H
MASS    7 HR1    1.00800 H ! his he1, (+) his HG,HD2
MASS    8 HR2    1.00800 H ! (+) his HE1
MASS    9 HR3    1.00800 H ! neutral his HG, HD2
MASS   10 HS     1.00800 H ! thiol hydrogen
MASS   11 HE1    1.00800 H ! for alkene; RHC=CR
MASS   12 HE2    1.00800 H ! for alkene; H2C=CR
MASS   20 C      12.01100 C ! carbonyl C, peptide backbone
MASS   21 CA     12.01100 C ! aromatic C
MASS   22 CT1    12.01100 C ! aliphatic sp3 C for CH
MASS   23 CT2    12.01100 C ! aliphatic sp3 C for CH2
MASS   24 CT3    12.01100 C ! aliphatic sp3 C for CH3
```

When specifying the connectivity of a chain of residues in a protein, it is necessary to refer to atoms in the previous or succeeding residue. The CHARMM topology file declares those atom types that will be referenced in adjoining residues as such:

```
DECL -CA
DECL -C
DECL -O
DECL +N
DECL +HN
DECL +CA
```



The first and last residues of a chain obviously have different connectivity from those in the center, since they have one fewer neighbor. This is handled by applying *patch residues*, normally referred to as *patches*, to the terminal residues. As will be seen, any residue definition can specify the patch to be applied when it is the first or last residue in a segment. However, a default set is declared for the entire file, as in the following where the default patch is NTER for the first residue of a segment and CTER for the last:

```
DEFA FIRS NTER LAST CTER
```

While the covalent bond connectivity between atoms must necessarily be provided by the topology file, enumerating all of the required angles and dihedrals would be tedious and error-prone, as well as enormously complicated since every combination of residues joined by a peptide bond would require a different set. Therefore, angles and dihedrals are automatically generated for every pair or triple of connected bonds when a segment is built. This autogeneration may be enabled or disabled on a per-segment basis as it should never be used on segments of water, but the default is defined in the topology file:

```
AUTO ANGLES DIHE
```

We are now ready for the actual residue definitions, beginning with alanine, as shown below. A residue is indicated by the RESI statement with the residue name (ALA) and total charge (0.00). Next are listed all of the atoms in the residue in ATOM statements with the atom name (N, HN, CA), type (NH1, H, CT1), and partial charge (-0.47, 0.31, 0.07). The GROUP statements, dividing the atoms into integer-charge groups, are not used by NAMD and should not be confused with the em hydrogen groups, each a non-hydrogen atom and all hydrogens bonded to it, that NAMD uses to accelerate distance-testing for nonbonded calculations.

```
RESI ALA          0.00
GROUP
ATOM N    NH1    -0.47 !    |
ATOM HN   H      0.31 !   HN-N
ATOM CA   CT1    0.07 !    |    HB1
ATOM HA   HB     0.09 !    |    /
GROUP          !   HA-CA--CB-HB2
ATOM CB   CT3   -0.27 !    |    \
ATOM HB1  HA     0.09 !    |    HB3
ATOM HB2  HA     0.09 !    O=C
ATOM HB3  HA     0.09 !    |
GROUP          !
ATOM C    C      0.51
ATOM O    O     -0.51
```

The ALA residue continues by defining connectivity, with each BOND statement followed by a list of pairs of atoms to be connected with bonds. The

DOUBLE statement is a synonym for BOND and does not affect the resulting PSF file. Observe that the atom C is bonded to +N, the N of the following residue. A bond between N and -C will be provided by the preceding residue. The order of bonds, or of the atoms within a bond, is not significant.

```
BOND CB CA N HN N CA
BOND C CA C +N CA HA CB HB1 CB HB2 CB HB3
DOUBLE O C
```

As noted above, the angle and dihedral terms will be autogenerated and are therefore not listed for this residue. The less common improper dihedrals (normally just called impropers), however, must be listed explicitly. In this case there are two impropers, which maintain the planarity of the peptide bonds. As with dihedrals, the order of atoms within an improper may be reversed. As shown below, impropers are specified by the IMPR statement followed by sets of four atoms, with the central atom to which the other three are bonded typically listed first.

```
IMPR N -C CA HN C CA +N O
```

Explicit hydrogen bond terms are no longer present in the CHARMM force field and are therefore not calculated by NAMD. The DONOR and ACCEPTOR statements, shown below, specify pairs of atoms eligible to form hydrogen bonds. The psfgen module in VMD ignores these statements and does not incorporate hydrogen bonding information into the PSF file.

```
DONOR HN N
ACCEPTOR O C
```

Finally in the residue definition are the internal coordinate IC statements. For each set of four atoms 1 2 3 4, the IC specifies in order the bond length 1-2, the angle 1-2-3, the dihedral 1-2-3-4, the angle 2-3-4, and the bond length 3-4. With this set of data, the position of atom 1 may be determined based on the positions of atoms 2-4, and the position of atom 4 may be determined from the positions of atoms 1-3, allowing the recursive generation of coordinates for all atoms in the structure based on a three-atom seed. Improper IC statements are indicated by a \* preceding the third atom, the atom to which the other three are bonded, as in 1 2 \*3 4. The order of atoms in an IC statement is different from that of an IMPR statement, and values provide the length 1-3, the angle 1-3-2, the dihedral 1-2-3-4, the angle 2-3-4, and the length 3-4.

```
IC -C CA *N HN 1.3551 126.4900 180.0000 115.4200 0.9996
IC -C N CA C 1.3551 126.4900 180.0000 114.4400 1.5390
IC N CA C +N 1.4592 114.4400 180.0000 116.8400 1.3558
IC +N CA *C O 1.3558 116.8400 180.0000 122.5200 1.2297
IC CA C +N +CA 1.5390 116.8400 180.0000 126.7700 1.4613
IC N C *CA CB 1.4592 114.4400 123.2300 111.0900 1.5461
IC N C *CA HA 1.4592 114.4400 -120.4500 106.3900 1.0840
```

```

IC C    CA    CB    HB1    1.5390 111.0900 177.2500 109.6000 1.1109
IC HB1  CA    *CB  HB2    1.1109 109.6000 119.1300 111.0500 1.1119
IC HB1  CA    *CB  HB3    1.1109 109.6000 -119.5800 111.6100 1.1114

```

It was mentioned above that special treatment is required for the first or last residue in a chain, and that this is implemented as a patch. The following is the default first-residue patch for proteins, NTER. The syntax is almost identical to a normal residue, but the ATOM statements may refer to either add a new atom (HT1, HT2, HT3) or modify the type and charge of an existing atom of the given name (N, CA, HA). The DELETE statement operates as one would expect, removing the atom (HN) and any bonds that include it.

```

PRES NTER          1.00 ! standard N-terminus
GROUP              ! use in generate statement
ATOM N    NH3     -0.30 !
ATOM HT1  HC      0.33 !      HT1
ATOM HT2  HC      0.33 !      (+)/
ATOM HT3  HC      0.33 ! --CA--N--HT2
ATOM CA   CT1     0.21 ! | \
ATOM HA   HB      0.10 !  HA  HT3
DELETE ATOM HN
BOND HT1 N HT2 N HT3 N
DONOR HT1 N
DONOR HT2 N
DONOR HT3 N
IC HT1  N    CA   C    0.0000 0.0000 180.0000 0.0000 0.0000
IC HT2  CA   *N   HT1  0.0000 0.0000 120.0000 0.0000 0.0000
IC HT3  CA   *N   HT2  0.0000 0.0000 120.0000 0.0000 0.0000

```

The comment “use in generate statement” indicates that the NTER patch is used during segment generation, and is applied before angles and dihedrals are generated. An example of the other type of patch, those applied after segment generation, is the LINK patch given below. The patch statement requires, in this case, two residues as arguments, the first being the last residue of a segment generated without the CTER patch and the second being the first residue of a segment generated without the NTER patch (the comments have the N and C termini reversed). The numbers 1 and 2 preceding atom names indicate which argument residue the named atom belongs to. Since angles and dihedrals will not be regenerated, they are enumerated.

```

PRES LINK          0.00 ! linkage for IMAGES or for joining segments
                   ! 1 refers to previous (N terminal)
                   ! 2 refers to next (C terminal)
                   ! use in a patch statement

BOND 1C 2N
ANGLE 1C 2N 2CA 1CA 1C 2N
ANGLE 1O 1C 2N 1C 2N 2HN
DIHE 1C 2N 2CA 2C 1C 2N 2CA 2HA 1C 2N 2CA 2CB
DIHE 1HA 1CA 1C 2N 1N 1CA 1C 2N 1CB 1CA 1C 2N

```

```

DIHE 1CA 1C 2N 2HN 1CA 1C 2N 2CA
DIHE 1O 1C 2N 2HN 1O 1C 2N 2CA
IMPR 2N 1C 2CA 2HN 1C 1CA 2N 1O
IC 1N 1CA 1C 2N 0.0000 0.0000 180.0000 0.0000 0.0000
IC 2N 1CA *1C 1O 0.0000 0.0000 180.0000 0.0000 0.0000
IC 1CA 1C 2N 2CA 0.0000 0.0000 180.0000 0.0000 0.0000
IC 1C 2N 2CA 2C 0.0000 0.0000 180.0000 0.0000 0.0000
IC 1C 2CA *2N 2HN 0.0000 0.0000 180.0000 0.0000 0.0000

```

These types of patches are used to alter protonation states (ASPP, GLUP, HS2), create disulphide bonds (DISU), attach HEME groups (PHEM) and their ligands (PLO2, PLIG), and even remove unwanted autogenerated angles (FHEM).

The following is the complete residue definition for GLY, the smallest of the amino acids. Compare GLY to the ALA residue dissected above.

```

RESI GLY          0.00
GROUP
ATOM N    NH1    -0.47 !    |
ATOM HN   H       0.31 !    N-H
ATOM CA   CT2    -0.02 !    |
ATOM HA1  HB      0.09 !    |
ATOM HA2  HB      0.09 ! HA1-CA-HA2
GROUP          !    |
ATOM C    C       0.51 !    |
ATOM O    O      -0.51 !    C=O
              !    |
BOND N HN  N  CA  C CA
BOND C +N  CA HA1 CA HA2
DOUBLE O  C
IMPR N -C  CA HN  C CA  +N O
DONOR HN N
ACCEPTOR O C
IC -C  CA  *N  HN  1.3475 122.8200 180.0000 115.6200 0.9992
IC -C  N   CA  C   1.3475 122.8200 180.0000 108.9400 1.4971
IC N   CA  C   +N  1.4553 108.9400 180.0000 117.6000 1.3479
IC +N  CA  *C  O   1.3479 117.6000 180.0000 120.8500 1.2289
IC CA  C   +N  +CA 1.4971 117.6000 180.0000 124.0800 1.4560
IC N   C   *CA HA1 1.4553 108.9400 117.8600 108.0300 1.0814
IC N   C   *CA HA2 1.4553 108.9400 -118.1200 107.9500 1.0817
PATCHING FIRS GLYP

```

Unlike ALA and most protein residues, in which CA is bonded to HA and CB, in GLY it is bonded to a pair of hydrogens, HA1 and HA2. Also, the hydrogen bonded to N is named H rather than HN. For these reasons, the default NTER patch cannot be applied to GLY and the PATCHING statement is used to change the default first-residue patch for GLY to GLYP. Similarly, the LINK patch above cannot be used for GLY residues, so the additional patches LIG1, LIG2, and LIG3 are provided to link GLY to non-GLY, non-GLY to GLY, and GLY to GLY.

Water is also defined as a residue, as shown below, but some care is required for its use. A third bond, not required by NAMD, is included to allow CHARMM to make the water molecule rigid. This ring structure would confuse angle and dihedral autogeneration, so segments of water must be generated with autogeneration disabled, and therefore an explicit angle is also included. If this third bond is removed from the water topology, then autogeneration must still be disabled to avoid duplicating the angle, unless the angle is also removed.

```
RESI TIP3          0.000 ! tip3p water model, generate using noangle nodihedral
GROUP
ATOM OH2  OT      -0.834
ATOM H1   HT       0.417
ATOM H2   HT       0.417
BOND OH2 H1 OH2 H2 H1 H2    ! the last bond is needed for shake
ANGLE H1 OH2 H2             ! required
ACCEPTOR OH2
PATCHING FIRS NONE LAST NONE
```

The residue definitions for the ions below are exceedingly simple.

```
RESI SOD          1.00 ! Sodium Ion
GROUP
ATOM SOD  SOD     1.00
PATCHING FIRST NONE LAST NONE

RESI CLA          -1.00 ! Chloride Anion
GROUP
ATOM CLA  CLA     -1.00
PATCHING FIRST NONE LAST NONE
```

We have discussed only those parts of the topology files associated with proteins and solvent. There is much additional information regarding proteins, not to mention lipids and nucleic acids, included in the comments in the topology files themselves.



!empirical potential for molecular modeling and dynamics Studies of  
!proteins. Journal of Physical Chemistry B, 1998, 102, 3586-3616.

The first set of entries in the parameter file are those for bonds, indicated by the BONDS keyword. Each entry consists of a pair of atom types, a spring constant, and an equilibrium length. The bond potential function is  $K_b(b-b_0)^2$ , where  $b$  is the bond length in Angstroms. Bonds are a stiff degree of freedom in biomolecules, so the energy function is only accurate for values near the equilibrium length. Entries are present for every type of bond present in the topology file. The beginning of the bonds section is shown below:

```

BONDS
!
!V(bond) = Kb(b - b0)**2
!
!Kb: kcal/mole/A**2
!b0: A
!
!atom type Kb          b0
!
C    C    600.000      1.3350 ! ALLOW ARO HEM
          ! Heme vinyl substituent (KK, from propene (JCS))
CA   CA   305.000      1.3750 ! ALLOW  ARO
          ! benzene, JES 8/25/89
CE1  CE1  440.000      1.3400  !
          ! for butene; from propene, yin/adm jr., 12/95
CE1  CE2  500.000      1.3420  !
          ! for propene, yin/adm jr., 12/95
CE1  CT2  365.000      1.5020  !
          ! for butene; from propene, yin/adm jr., 12/95
CE1  CT3  383.000      1.5040  !
          ! for butene, yin/adm jr., 12/95
CE2  CE2  510.000      1.3300  !
          ! for ethene, yin/adm jr., 12/95

```

The CHARMM potential function is designed to confuse physicists, as the form is  $K_b(b-b_0)^2$ , rather than the traditional  $\frac{1}{2}k_b(b-b_0^2)$ , and therefore the  $K_b$  given in the parameter files is half the value of a traditional spring constant. Also, while the units kcal/mol for energy, Angstroms for length, atomic masses, electron charges, and either fs or ps for time may be convenient for input and output, tortuous unit conversions are required to express the equations of motion.

The next section gives parameters for every type of angle present in the topology file, indicated by the ANGLES keyword. The angle potential function is  $K_\theta(\theta-\theta_0)^2$ , where  $\theta$  is the measure of the angle in degrees. Angles are a stiff degree of freedom in biomolecules as well, so the energy function is only accurate for values near the equilibrium angle. Since angles are formed from combinations of bonds, there are many more types of angles than types of bonds. Each entry consists of three atom types, a spring constant, and an equilibrium

angle. A small minority of entries also contain Urey-Bradley parameters, which are a spring constant and equilibrium length for a bond-like term between the first and third atoms in the angle. The beginning of the angles section is shown below, with a Urey-Bradley term in the first entry only:

```

ANGLES
!
!V(angle) = Ktheta(Theta - Theta0)**2
!
!V(Urey-Bradley) = Kub(S - S0)**2
!
!Ktheta: kcal/mole/rad**2
!Theta0: degrees
!Kub: kcal/mole/A**2 (Urey-Bradley)
!S0: A
!
!atom types      Ktheta      Theta0      Kub        S0
!
CA  CA  CA      40.000      120.00      35.00      2.41620 ! ALLOW  ARD
! JES 8/25/89
CE1 CE1 CT3      48.00      123.50      !
! for 2-butene, yin/adm jr., 12/95
CE1 CT2 CT3      32.00      112.20      !
! for 1-butene; from propene, yin/adm jr., 12/95
CE2 CE1 CT2      48.00      126.00      !
! for 1-butene; from propene, yin/adm jr., 12/95
CE2 CE1 CT3      47.00      125.20      !
! for propene, yin/adm jr., 12/95

```

The next section gives parameters for every type of dihedral present in the topology file; there are even more dihedrals than there are angles. Since dihedrals represent the energy of rotation around a covalent bond, which is the source of most conformational flexibility in biomolecules, they must provide a smooth energy for 360 degrees. This is done in most cases with a single sinusoid,  $K_\chi(1 + \cos(n(\chi - \delta)))$  where  $\chi$  is the angle between the plane containing the first three atoms in the dihedral and the plane containing the last three. The “multiplicity”  $n$  is typically 1, 2, or 3, although for a small number of cases two or three terms with different values of  $n$  are provided for the same atom types. You may observe in the excerpts below that the dihedral spring constants are one to two orders of magnitude lower than for angles, with an order of magnitude difference between flexible and inflexible dihedrals.

```

DIHEDRALS
!
!V(dihedral) = Kchi(1 + cos(n(chi) - delta))
!
!Kchi: kcal/mole
!n: multiplicity
!delta: degrees

```



```

!
!atom types          Kchi    n    delta
!
C   CT1 NH1 C        0.2000  1   180.00 ! ALLOW PEP
! ala dipeptide update for new C VDW Rmin, adm jr., 3/3/93c
C   CT2 NH1 C        0.2000  1   180.00 ! ALLOW PEP
! ala dipeptide update for new C VDW Rmin, adm jr., 3/3/93c
C   N    CP1 C        0.8000  3    0.00 ! ALLOW PRO PEP
! 6-31g* AcProNH2, ProNH2, 6-31g**/3-21g AcProNHCH3 RLD 4/23/93
CA  CA   CA   CA      3.1000  2   180.00 ! ALLOW  ARO
! JES 8/25/89
CA  CPT  CPT  CA      3.1000  2   180.00 ! ALLOW  ARO
! JWK 05/14/91 fit to indole

```

Because of the large numbers of dihedral terms required to describe a complete protein, the wildcard atom type X is occasionally used. These parameters will be used in NAMD if a more specific match is not found elsewhere in the parameter file.

```

X   C    C    X        4.0000  2   180.00 ! ALLOW HEM
! Heme (6-liganded): substituents (KK 05/13/91)
X   C    NC2 X        2.2500  2   180.00 ! ALLOW  PEP POL ARO
! 9.0->2.25 GUANIDINIUM (KK)
X   CD   OH1 X        2.0500  2   180.00 ! ALLOW  PEP POL ARO ALC
! adm jr, 10/17/90, acetic acid C-Oh rotation barrier
X   CD   OS  X        2.0500  2   180.00 ! ALLOW  PEP POL
! adm jr. 3/19/92, from lipid methyl acetate
X   CE1  CE1 X        5.2000  2   180.00 !
! for butene, yin/adm jr., 12/95
X   CE2  CE2 X        4.9000  2   180.00 !
! for ethene, yin/adm jr., 12/95

```

The final bond-like terms in the parameter file are impropers, which are used exclusively and explicitly in the molecular topology to maintain planarity. As such, the harmonic form  $K_\psi(\psi - \psi_0)^2$  with a large spring constant and  $\psi_0$  typically zero is used to restrain deformations among an atom and three atoms bonded to it. As with dihedrals,  $\psi$  is angle between the plane containing the first three atoms and the plane containing the last three. Notice below that wildcard atom types occur in the second and third positions, rather than the first and fourth as in dihedrals.

```

IMPROPER
!
!V(improper) = Kpsi(psi - psi0)**2
!
!Kpsi: kcal/mole/rad**2
!psi0: degrees
!note that the second column of numbers (0) is ignored
!

```

```

!atom types          Kpsi          psi0
!
CPB CPA NPH CPA    20.8000          0    0.0000 ! ALLOW HEM
! Heme (6-liganded): porphyrin macrocycle (KK 05/13/91)
CPB X  X  C     90.0000          0    0.0000 ! ALLOW HEM
! Heme (6-liganded): substituents (KK 05/13/91)
CT2 X  X  CPB   90.0000          0    0.0000 ! ALLOW HEM
! Heme (6-liganded): substituents (KK 05/13/91)
CT3 X  X  CPB   90.0000          0    0.0000 ! ALLOW HEM
! Heme (6-liganded): substituents (KK 05/13/91)
HA  C  C  HA    20.0000          0    0.0000 ! ALLOW PEP POL ARO
! Heme vinyl substituent (KK, from propene (JCS))

```

Finally we come to the nonbonded interaction parameters. The NONBONDED statement includes a list of parameters which are used as defaults by the program CHARMM, but are ignored by NAMD. Those shown below correspond to the NAMD settings exclude scaled1-4, switching on, pairlistdist 14.0, cutoff 12.0, switchdist 10.0, dielectric 1.0, and 1-4scaling 1.0:

```

NONBONDED nbxmod 5 atom cdie1 shift vatom vdistance vswitch -
cutnb 14.0 ctofnb 12.0 ctonnb 10.0 eps 1.0 e14fac 1.0 wmin 1.5
!adm jr., 5/08/91, suggested cutoff scheme

```

Recall that the partial charge of each atom is specified in the topology and PSF files and is independent of the atom type. Therefore the only type-based parameters are for the van der Waals interactions, which are represented by the classic Lennard-Jones potential (expressed in the somewhat unconventional form)  $\epsilon[(r_{min}/r)^{12} - 2(r_{min}/r)^6]$ . Observe that at  $r = r_{min}$  the force is zero and the energy is  $-\epsilon$ . Rather than providing a different value of epsilon could be provided for every possible combination of atom types, only one value is provided per type and inter-type interactions are calculated using the sum of the radii  $r_{min}/2$  and the geometric mean of the well-depths  $\epsilon$ . By convention, the  $\epsilon$  values are negative in the parameter file, as seen here:

```

!
!V(Lennard-Jones) = Eps,i,j[(Rmin,i,j/ri,j)**12 - 2(Rmin,i,j/ri,j)**6]
!
!epsilon: kcal/mole, Eps,i,j = sqrt(eps,i * eps,j)
!Rmin/2: A, Rmin,i,j = Rmin/2,i + Rmin/2,j
!
!atom ignored   epsilon      Rmin/2   ignored   eps,1-4      Rmin/2,1-4
!
C      0.000000  -0.110000    2.000000 ! ALLOW   PEP POL ARO
! NMA pure solvent, adm jr., 3/3/93
CA     0.000000  -0.070000    1.992400 ! ALLOW   ARO
! benzene (JES)
CC     0.000000  -0.070000    2.000000 ! ALLOW   PEP POL ARO
! adm jr. 3/3/92, acetic acid heat of solvation
CD     0.000000  -0.070000    2.000000 ! ALLOW   POL

```

```

! adm jr. 3/19/92, acetate a.i. and dH of solvation
CE1  0.000000  -0.068000   2.090000  !
! for propene, yin/adm jr., 12/95
CE2  0.000000  -0.064000   2.080000  !
! for ethene, yin/adm jr., 12/95
CM   0.000000  -0.110000   2.100000  ! ALLOW HEM
! Heme (6-liganded): CO ligand carbon (KK 05/13/91)

```

When the scaled1-4 exclusion policy is used (as it should with the CHARMM force field) nonbonded interactions of atoms separated by three bonds (i.e., atoms 1 and 4 in the chain 1-2-3-4) are modified. Even if the scaling factor for electrostatics is 1.0 (as it should be for modern CHARMM force fields), special modified van der Waals parameters are used for 1-4 pairs of atoms for which they are specified, as in the examples below.

```

!atom  ignored    epsilon    Rmin/2  ignored  eps,1-4    Rmin/2,1-4
!
CP1   0.000000  -0.020000   2.275000  0.000000  -0.010000   1.900000  ! ALLOW  ALI
! alkane update, adm jr., 3/2/92
CP2   0.000000  -0.055000   2.175000  0.000000  -0.010000   1.900000  ! ALLOW  ALI
! alkane update, adm jr., 3/2/92
CP3   0.000000  -0.055000   2.175000  0.000000  -0.010000   1.900000  ! ALLOW  ALI
! alkane update, adm jr., 3/2/92

```

The parameter file ends with a reference to parameters for explicit hydrogen bond energy terms. These are obsolete, no longer present in the CHARMM force field, and therefore not implemented by NAMD.

```

HBOND CUTHB 0.5 ! If you want to do hbond analysis (only), then use
! READ PARAM APPEND CARD
! to append hbond parameters from the file: par_hbond.inp

```

END

For information on how the parameters have been derived, you must consult the corresponding publications referenced in the parameter files themselves or listed on the MacKerell web site <http://www.pharmacy.umaryland.edu/faculty/amackere/research.html>.

## E NAMD Configuration Files

The NAMD configuration file (also called a config file, .conf file, or .namd file) is given to NAMD on the command line and specifies virtually everything about the simulation to be done. The only exceptions are details relating to the parallel execution environment, which vary between platforms. Therefore, the config file should be portable between machines, platforms, or numbers of processors in a run, as long as the referenced input files are available.

As a convenience, on startup NAMD will switch to the directory that contains the config file, so that all file paths are relative to that directory. NAMD also accepts multiple config files on the command line, but changes directories before parsing each file, so it is best to keep everything in the same directory when using multiple config files.

NAMD parses its configuration file using the Tcl scripting language, which you should be familiar with if you have done any serious work using VMD. However, to remain compatible with config files from earlier version of NAMD which did not include Tcl, several compromises have been made. Tcl is a case sensitive language, so the `$temp` and `$TEMP` are different variables. NAMD's config file options are implemented as case-insensitive Tcl commands, however, so `temperature 300`, `Temperature 300`, or even `tEmPerATURe 300` will all have the same effect.

NAMD config files were historically order-independent, and the whole file was parsed before any files were read or calculations done. Now that NAMD incorporates more advanced scripting, the config file is parsed up to the point where one of the case-sensitive “action” commands such as `minimize` or `run` are encountered; at this point the input files are read and real calculation begins. Beyond this point, the config file is order-dependent and only these action commands and a subset of configuration options may be given. Also, at this point any comments must follow the rules of Tcl, using a semicolon to end a command before using `#` to start a comment on the same line; it is good practice to follow these rules everywhere in the file.

We will now walk through a NAMD configuration file template that contains options for common simulation methods in NAMD. A version of this file should be available in the tutorial materials as `sample.conf`.

First we specify the files that contain the molecular structure and initial conditions. A `coordinates` PDB file is always required, even if the actual coordinates will come from a binary coordinates file.

```
structure      mypsf.psf
coordinates    mypdb.pdb
```

Setting the Tcl variable `$temperature` makes it easy to change the target temperature for many options.

```
set temperature 310 ;# target temperature used several times below
```

If we are starting from scratch, we'll use the coordinates from the PDB file and take velocities randomly from a Boltzmann distribution, using the `$temperature` variable.

```
# starting from scratch
temperature      $temperature      ;# initialize velocities randomly
```

Otherwise we'll read in the binary output files of the previous run. We use the Tcl variable `$inputname` to avoid errors in typing the input file names, since we will end up copying and modifying this file several times during the course of a long simulation. The `extendedSystem` file holds the periodic cell dimensions that are needed to continue after a constant pressure simulation. In order to start numbering timesteps with the final step of the previous run, we use `firsttimestep` to specify the timestep on which the input coordinates and velocities were written. The number of steps calculated in this run will be `numsteps - firsttimestep`.

```
# continuing a run
set inputname    myinput            ;# only need to edit this in one place!
binCoordinates   $inputname.coor    ;# coordinates from last run (binary)
binVelocities    $inputname.vel     ;# velocities from last run (binary)
extendedSystem   $inputname.xsc     ;# cell dimensions from last run
firsttimestep    50000              ;# last step of previous run
numsteps         100000             ;# run stops when this step is reached
```

The `outputName` prefix will be used to create all of the trajectory (`.dcd`, `.xst`), output (`.coor`, `vel`, `.xsc`), and restart (`.restart.coor`, `.restart.vel`, `.restart.xsc`) files generated by this run. Be careful that this is different from `$myinput`, or the job will overwrite its input files!

```
outputName       myoutput          ;# base name for output from this run
```

We will write restart files, coordinate trajectory `.dcd` files, and extended system (periodic cell) trajectory `.xst` files at regular intervals.

```
restartfreq      500               ;# 500 steps = every 1ps
dcdfreq          500
xstFreq         500
```

The default is to print energies every steps, but that makes for a very long file, so we'll cut it down to every 100 steps. We'll also enable the printing of performance information every 1000 steps.

```
outputEnergies   100               ;# 100 steps = every 0.2 ps
outputTiming     1000              ;# shows time per step and time to completion
```

Next are the parameter file itself and the options that control the nonbonded potential functions. These are mostly specified by the CHARMM force field, but the cutoff and other distances may be shortened when full electrostatics are used; Just be sure that you adjust all of the distances together. The `pairlistdist` can be made longer if you see warnings in the NAMD output, but making it bigger than necessary will reduce performance.

```

# Force-Field Parameters
paraTypeCharmm      on
parameters          par_all27_prot_lipid.inp

# These are specified by CHARMM
exclude             scaled1-4
1-4scaling         1.0
switching          on

# You have some freedom choosing the cutoff
cutoff             12. ;# may use smaller, maybe 10., with PME
switchdist        10. ;# cutoff - 2.

# Promise that atom won't move more than 2A in a cycle
pairlistdist      14. ;# cutoff + 2.

```

The chosen `stepspercycle` is related to `pairlistdist` above, as pairlists are generated at the beginning of each cycle and are assumed to contain every atom that passes within the cutoff distance during entire cycle. Warnings will appear in the output if `pairlistdist` is too short (or `stepspercycle` is too large). The length of the simulation must be an integer number of cycles.

```
stepspercycle      10 ;# redo pairlists every ten steps
```

The integration timestep is normally limited to 1 fs. This can be extended to 2 fs by fixing the length of all bonds in the molecule involving hydrogen atoms via `rigidBonds all`, which also makes water molecules completely rigid. To only make water rigid, use `rigidBonds water` and a 1 fs timestep. Nonbonded forces must be calculated at least every 2 fs, which in this example is every step. Full electrostatics forces (from particle mesh Ewald, discussed below) are evaluated every other step in this example. `nonbondedFreq` and `fullElectFrequency` must evenly divide `stepspercycle`.

```

# Integrator Parameters
timestep          2.0 ;# 2fs/step
rigidBonds       all ;# needed for 2fs steps
nonbondedFreq    1 ;# nonbonded forces every step
fullElectFrequency 2 ;# PME only every other step

```

Langevin dynamics balances friction with random noise to drive each atom in the system towards a target temperature. The following parameters are good for equilibration, but a `langevinDamping` as low as 1. would be sufficient for simulation. Higher damping may be necessary if, for example, work done on the system by steering forces is driving up the temperature.

```

# Constant Temperature Control
langevin         on ;# langevin dynamics
langevinDamping  5. ;# damping coefficient of 5/ps
langevinTemp     $temperature ;# random noise at this level
langevinHydrogen no ;# don't couple bath to hydrogens

```

If the simulation will use periodic boundary conditions, they are specified as shown below. Like the `temperature` option, these should only be given when starting a simulation from scratch, since the basis vectors will fluctuate during constant pressure simulation and updated values need to be read via `extendedSystem` from a `.xsc` file.

```
# Periodic Boundary conditions
cellBasisVector1 31.2 0. 0. ;# vector to the next image
cellBasisVector2 0. 44.8 0.
cellBasisVector3 0. 0 51.3
cellOrigin 0. 0. 0. ;# the *center* of the cell
```

The cell origin is the one coordinate that does not change during constant pressure simulation. This point plays a role in calculating the pressure contribution due to fixed atoms, harmonic restraints, and other “external” forces. To minimize pressure fluctuations, this point should be close to the center of the macromolecule of interest.

The origin also defines the center of the periodic cell to which coordinates will be wrapped on output if the following options below are used. With wrapping, some molecules will jump between sides of the cell in the trajectory file to yield the periodic image nearest to the origin. Without wrapping, molecules will have smooth trajectories, but water in the trajectory may appear to explode as individual molecules diffuse. Wrapping only affects output, not the correctness of the simulation.

```
wrapWater on ;# wrap water to central cell
wrapAll on ;# wrap other molecules too
wrapNearest off ;# use for non-rectangular cells
```

Particle mesh Ewald (PME) full electrostatics are more accurate and less expensive than larger cutoffs, and are recommended for most work. PME is only applicable to periodic simulations, and the user must specify a grid corresponding to the size of the periodic cell. PME grid dimensions should have small integer factors only and be greater than or equal to length of the basis vector.

```
#PME (for full-system periodic electrostatics)
PME yes
PMEGridSizeX 32 ;# 2^5, close to 31.2
PMEGridSizeY 45 ;# 3^2 * 5, close to 44.8
PMEGridSizeZ 54 ;# 2 * 3^3, close to 51.3
```

Constant pressure is recommended for periodic simulations. Using group-based pressure to control the periodic cell fluctuations is desirable because the atom-based pressure has more high-frequency noise. `useFlexibleCell` is useful for anisotropic systems such as membranes, allowing the height, length, and width of the cell to vary independently, possibly even fixing the lipid cross-sectional (x-y plane) area with `useConstantArea`. For a protein surrounded by water there is nothing to prevent the cell from becoming highly extended in one dimension, so it is better to choose `useFlexibleCell no` in this case.

```

# Constant Pressure Control (variable volume)
useGroupPressure      yes ;# needed for rigid bonds
useFlexibleCell       no  ;# no for water box, yes for membrane
useConstantArea       no  ;# no for water box, maybe for membrane

```

The actual parameters of the Nosé-Hoover Langevin piston method control the target pressure and the dynamical properties of the barostat. A “piston” with a longer period (i.e., larger mass) will better damp out fluctuations in the instantaneous pressure. Langevin dynamics is applied to the piston itself coupling it to a heat bath with a damping constant of  $1/\text{langevinPistonDecay}$ . We set `langevinPistonDecay` smaller than `langevinPistonPeriod` to ensure that harmonic oscillations in the periodic cell are overdamped.

```

langevinPiston        on
langevinPistonTarget  1.01325      ;# pressure in bar -> 1 atm
langevinPistonPeriod  100.         ;# oscillation period around 100 fs
langevinPistonDecay   50.          ;# oscillation decay time of 50 fs
langevinPistonTemp    $temperature ;# coupled to heat bath

```

When initially assembling a system it is sometimes useful to fix the protein while equilibrating water or lipids around it. These options read a PDB file containing flags for the atoms to fix. The number and order of atoms in the PDB file must exactly match that of the PSF and other input files.

```

fixedAtoms            on
fixedAtomsFile        myfixedatoms.pdb ;# flags are in this file
fixedAtomsCol         B                ;# set beta non-zero to fix an atom

```

The interactive MD features of NAMD and VMD allow you to connect to a running simulation to apply steering forces manually. These options affect performance, and should therefore not be used unless you are actually steering the simulation.

```

IMDon                on
IMDport              3000          ;# port number (enter it in VMD)
IMDfreq              1             ;# send every 1 frame
IMDwait              no           ;# wait for VMD to connect before running?

```

Now we minimize the system to eliminate bad initial contacts, reinitialize the velocities to the desired target temperature (since minimization sets velocities to zero), and run for 100 ps. We could accomplish the same thing with two different NAMD runs using the `numsteps` and `minimization` options. Scripting commands such as those below override `numsteps`.

```

minimize              1000         ;# lower potential energy for 1000 steps
reinitvels           $temperature ;# since minimization zeros velocities
run 50000 ;# 100ps

```

Naturally, these are not all of the configuration options accepted by NAMD, but only a rapid introduction to those you are most likely to encounter. Documentation for these and many other options can be found in the Users Guide.



## F NAMD Standard Output

When NAMD runs, important information on the progress of the simulation is written to standard output, appearing on the console unless output from NAMD is redirected to a file. This output is designed to be easy for both humans and programs to interpret.

At the beginning of the file, the version of NAMD being used is presented, along with other useful information, some of which is reported back to the developers over the network to help us track NAMD's popularity.

```
Info: NAMD 2.5b2ss03 for Linux-i686-Clustermatic
Info:
Info: Please visit http://www.ks.uiuc.edu/Research/namd/
Info: and send feedback or bug reports to namd@ks.uiuc.edu
Info:
Info: Please cite Kale et al., J. Comp. Phys. 151:283-312 (1999)
Info: in all publications reporting results obtained with NAMD.
Info:
Info: Built Fri May 30 13:09:06 CDT 2003 by jim on umbriel
Info: Sending usage information to NAMD developers via UDP.
Info: Sent data is: 1 NAMD 2.5b2ss03 Linux-i686-Clustermatic 47 umbriel jim
Info: Running on 47 processors.
```

This is followed by a long list of the various configuration options and summaries of information extracted from the parameter and structure files. This is mostly self-explanatory and just restates the config file options, or the defaults for options not specified.

NAMD goes through several phases to setup data structures for the actual simulation, reporting memory usage (for the first process only!). Patches are the cells that NAMD uses to spread atoms across a parallel simulation, although the size of the patch grid is independent of the number of processors being used.

```
Info: Entering startup phase 0 with 6668 kB of memory in use.
Info: Entering startup phase 1 with 6668 kB of memory in use.
Info: Entering startup phase 2 with 10212 kB of memory in use.
Info: Entering startup phase 3 with 10212 kB of memory in use.
Info: PATCH GRID IS 4 (PERIODIC) BY 4 (PERIODIC) BY 4 (PERIODIC)
Info: REMOVING COM VELOCITY -0.0765505 0.00822415 -0.00180344
Info: LARGEST PATCH (31) HAS 408 ATOMS
Info: Entering startup phase 4 with 13540 kB of memory in use.
Info: Entering startup phase 5 with 13540 kB of memory in use.
Info: Entering startup phase 6 with 13540 kB of memory in use.
Info: Entering startup phase 7 with 13540 kB of memory in use.
Info: COULOMB TABLE R-SQUARED SPACING: 0.0625
Info: COULOMB TABLE SIZE: 705 POINTS
Info: NONZERO IMPRECISION IN COULOMB TABLE: 4.03897e-28 (692) 1.00974e-27 (692)
Info: NONZERO IMPRECISION IN COULOMB TABLE: 2.5411e-21 (701) 5.92923e-21 (701)
Info: Entering startup phase 8 with 13540 kB of memory in use.
Info: Finished startup with 13540 kB of memory in use.
```

Energies, along with temperatures and pressures, are printed as a single line with a unique prefix for easy processing with utilities such as `grep` and `awk`. An `ETITLE` line is printed once for every ten `ENERGY` lines. The spacing of the fields, however, is designed to be easy to scan on an 80-column display, as seen below:

| <code>ETITLE:</code> | <code>TS</code>          | <code>BOND</code>      | <code>ANGLE</code>       | <code>DIHED</code>       | <code>IMPRP</code>      |
|----------------------|--------------------------|------------------------|--------------------------|--------------------------|-------------------------|
|                      | <code>ELECT</code>       | <code>VDW</code>       | <code>BOUNDARY</code>    | <code>MISC</code>        | <code>KINETIC</code>    |
|                      | <code>TOTAL</code>       | <code>TEMP</code>      | <code>TOTAL2</code>      | <code>TOTAL3</code>      | <code>TEMPAVG</code>    |
|                      | <code>PRESSURE</code>    | <code>GPRESSURE</code> | <code>VOLUME</code>      | <code>PRESSAVG</code>    | <code>GPRESSAVG</code>  |
| <code>ENERGY:</code> | <code>1000</code>        | <code>0.0000</code>    | <code>0.0000</code>      | <code>0.0000</code>      | <code>0.0000</code>     |
|                      | <code>-97022.1848</code> | <code>9595.3175</code> | <code>0.0000</code>      | <code>0.0000</code>      | <code>14319.5268</code> |
|                      | <code>-73107.3405</code> | <code>300.2464</code>  | <code>-73076.6148</code> | <code>-73084.1411</code> | <code>297.7598</code>   |
|                      | <code>-626.5205</code>   | <code>-636.6638</code> | <code>240716.1374</code> | <code>-616.5673</code>   | <code>-616.6619</code>  |

Energy values are in kcal/mol. This example is from a simulation of pure, rigid water, so the `BOND`, `ANGLE`, `DIHED`, and `IMPRP` terms are all zero. `BOUNDARY` energy is from spherical boundary conditions and harmonic restraints, while `MISC` energy is from external electric fields and various steering forces. `TOTAL` is the sum of the various potential energies, and the `KINETIC` energy. `TOTAL2` uses a slightly different kinetic energy that is better conserved during equilibration in a constant energy ensemble. `TOTAL3` is another variation with much smaller short-time fluctuations that is also adjusted to have the same running average as `TOTAL2`. Defects in constant energy simulations are much easier to spot in `TOTAL3` than in `TOTAL` or `TOTAL2`.

Pressure values are in bar. Temperature values are in Kelvin. `PRESSURE` is the pressure calculated based on individual atoms, while `GPRESSURE` incorporates hydrogen atoms into the heavier atoms to which they are bonded, producing smaller fluctuations. The `TEMPAVG`, `PRESSAVG`, and `GPRESSAVG` are the average of temperature and pressure values since the previous `ENERGY` output; for the first step in the simulation they will be identical to `TEMP`, `PRESSURE`, and `GPRESSURE`.

To estimate NAMD's performance for a long simulation, look for the "Benchmark time" lines printed in the first several hundred steps. These are a measure of NAMD's performance after startup and initial load balancing have been completed; this number should be very close to the long-time average.

```
Info: Benchmark time: 47 CPUs 0.0475851 s/step 0.275377 days/ns 13540 kB memory
```

Periodic performance output, if enabled, shows the CPU and wallclock time used by the first processor (not the aggregate of all processors in the simulation). The hours remaining are estimated based on performance since the last `TIMING` output.

```
TIMING: 1000 CPU: 18.35, 0.01831/step Wall: 50.1581, 0.0499508/step, 6.92374
hours remaining, 14244 kB of memory in use.
```

Lines such as these are printed whenever file output occurs.

OPENING COORDINATE DCD FILE  
WRITING COORDINATES TO DCD FILE AT STEP 1000

It is important to search the NAMD output for Warning and ERROR lines, which report unusual and possibly erroneous conditions. In the example below, the pairlist distance is too short or the cycle length is too long, possibly leading to reduced performance.

Warning: Pairlistdist is too small for 1 patches during timestep 17.  
Warning: Pairlists partially disabled; reduced performance likely.  
Warning: 20 pairlist warnings since previous energy output.

It is foolish to ignore warnings you do not understand!

## G Water Sphere tcl Script

```
# finds a center of mass of the molecule (ubiquitin), place a sphere of water
# around it.
```

```
# To run execute: vmd -dispdev text -e waterSphere.tcl
```

```
proc center_of_mass {selection} {

    # some error checking
    if {[ $\$selection$  num] <= 0} {
        error "center_of_mass: needs a selection with atoms"
    }
    # set the center of mass to 0
    set com [veczero]
    # set the total mass to 0
    set mass 0
    # [ $\$selection$  get {x y z}] returns the coordinates {x y z}
    # [ $\$selection$  get {mass}] returns the masses
    # so the following says "for each pair of {coordinates} and masses,
    # do the computation ..."
    foreach coord [ $\$selection$  get {x y z}] m [ $\$selection$  get mass] {
        # sum of the masses
        set mass [expr $mass + $m]
        # sum up the product of mass and coordinate
        set com [vecadd $com [vecscale $m $coord]]
    }
    # and scale by the inverse of the number of atoms
    if {$mass == 0} {
        error "center_of_mass: total mass is zero"
    }
    # The "1.0" can't be "1", since otherwise integer division is done
    return [vecscale [expr 1.0/$mass] $com]
}
```

```
#####
#                               MAIN PART STARTS HERE
#####
```

```
set psf      ubq.psf
set pdb      ubq.pdb
set box      ubq_box
set psfDrop  ubq_ws.psf
set pdbDrop  ubq_ws.pdb
```

```
package require psfgen
```

```
resetpsf
```

```

mol load psf $psf pdb $pdb
set sel [atomselect top all]
# find mass center
set center [center_of_mass $sel]
puts "center of mas is at $center"

foreach {xmass ymass zmass} $center { break }

set num0 9999
set Rmin 0.0

while {$num0 != 0} {
    set Rmin [expr $Rmin +1.0]
    set probSel [atomselect top "not (sqr(x-$xmass) + sqr(y-$ymass) +
sqr(z-$zmass) <= sqr($Rmin))"]
    set num0 [$probSel num]
    puts "$num0 $Rmin"
}

package require solvate

set xmin [expr $xmass -$Rmin]
set xmax [expr $xmass +$Rmin]

set ymin [expr $ymass -$Rmin]
set ymax [expr $ymass +$Rmin]

set zmin [expr $zmass -$Rmin]
set zmax [expr $zmass +$Rmin]

puts " $xmin $ymin $zmin $xmax $ymax $zmax"
#solvate $psf $pdb -o $box -minmax {{15.4399995804 12.8879995346
-0.365999996662} {46.125 45.2680015564 36.2509994507} }

set min "$xmin $ymin $zmin"
set max "$xmax $ymax $zmax"
set minmax [list $min $max]

solvate $psf $pdb -o $box -minmax $minmax

mol delete top

resetpsf

mol load psf ${box}.psf pdb ${box}.pdb

```

```

readpsf ${box}.psf
coordpdb ${box}.pdb

set selDel [atomselect top "not (sqr(x-$xmass) + sqr(y-$ymass) + sqr(z-$zmass)
<= sqr($Rmin))" ]
puts " not within [$selDel num]"

set testSel [atomselect top "not (sqr(x-$xmass) + sqr(y-$ymass) +
sqr(z-$zmass) <= sqr($Rmin)) and (not water)"]
puts " not within and not water [$testSel num]"

if { [$testSel num] != 0} {
    puts "ERROR: there are [$testSel num] non water molecules outside the shell"
    puts "EXIT"
    exit
}

set delList [$selDel get {segid resid}]

set delList [lsort -unique $delList]

foreach record $delList {
    foreach {segid resid} $record { break }
    delatom $segid $resid
}

writepsf $psfDrop
writepdb $pdbDrop

# remove temporary files generated by the script
file delete ${box}.psf ${box}.pdb combine.pdb combine.psf

puts "CENTER OF MASS IS AT: $center"
puts "SPHERE RADIUS:      $Rmin"

exit

```