NAMD User's Guide

Version 2.5b1

M. Bhandarkar, R. Brunner, C. Chipot, A. Dalke, S. Dixit, P. Grayson,
J. Gullingsrud, A. Gursoy, W. Humphrey, D. Hurwitz, N. Krawetz,
M. Nelson, J. Phillips, A. Shinozaki, G. Zheng, F. Zhu

April 16, 2002

Theoretical Biophysics Group
University of Illinois and Beckman Institute
405 N. Mathews
Urbana, IL 61801

Description

The NAMD *User's Guide* describes how to run and use the various features of the molecular dynamics program NAMD. This guide includes the capabilities of the program, how to use these capabilities, the necessary input files and formats, and how to run the program both on uniprocessor machines and in parallel.

NAMD Version 2.5b1

Authors: M. Bhandarkar, R. Brunner, C. Chipot, A. Dalke, S. Dixit, P. Grayson, J. Gullingsrud, A. Gursoy, W. Humphrey, D. Hurwitz, N. Krawetz, M. Nelson, J. Phillips, A. Shinozaki, G. Zheng, F. Zhu

Theoretical Biophysics Group, Beckman Institute, University of Illinois.

NAMD Molecular Dynamics Software
Non-Exclusive, Non-Commercial Use License

Introduction

The University of Illinois at Urbana-Champaign has created its molecular dynamics software, NAMD, developed by the Theoretical Biophysics Group ("TBG") at Illinois' Beckman Institute available free of charge for non-commercial use by individuals, academic or research institutions and corporations for in-house business purposes only, upon completion and submission of the online registration form available from the NAMD web site `http://www.ks.uiuc.edu/Research/namd/`.

Commercial use of the NAMD software, or derivative works based thereon, REQUIRES A COMMERCIAL LICENSE. Commercial use includes: (1) integration of all or part of the Software into a product for sale, lease or license by or on behalf of Licensee to third parties, or (2) distribution of the Software to third parties that need it to commercialize product sold or licensed by or on behalf of Licensee. The University of Illinois will negotiate commercial-use licenses for NAMD upon request. These requests can be directed to namd@ks.uiuc.edu

Registration

Individuals may register in their own name or with their institutional or corporate affiliations. Registration information must include name, title, and e-mail of a person with signature authority to authorize and commit the individuals, academic or research institution, or corporation as necessary to the terms and conditions of the license agreement.

All parts of the information must be understood and agreed to as part of completing the form. Completion of the form is required before software access is granted. Pay particular attention to the authorized requester requirements above, and be sure that the form submission is authorized by the duly responsible person.

Registration will be administered by the NAMD development team.

Contact Information

The best contact path for licensing issues is by e-mail to namd@ks.uiuc.edu or send correspondence to:

NAMD Team
Theoretical Biophysics Group
Beckman Institute
University of Illinois
405 North Mathews MC-251
Urbana, Illinois 61801 USA
FAX: (217) 244-6078

Contents

List of Figures

1 Introduction

NAMD is a parallel molecular dynamics program for UNIX platforms designed for high-performance simulations in structural biology. This document describes how to use NAMD, its features, and the platforms on which it runs. The document is divided into several sections:

Section 1 gives an overview of NAMD.

Section 2 lists the basics for getting started.

Section 3 describes NAMD file formats.

Section 4 explains PSF file generation with psfgen.

Section 5 lists basic simulation options.

Section 6 lists additional simulation options.

Section 7 provides hints for X-PLOR users.

Section 8 provides sample configuration files.

Section 9 gives details on running NAMD.

Section 10 gives details on installing NAMD.

We have attempted to make this document complete and easy to understand and to make NAMD itself easy to install and run. We welcome your suggestions for improving the documentation or code at `namd@ks.uiuc.edu`.

1.1 New features in version 2.5b1

Improved Constant Pressure Simulation and Coordinate Wrapping

Average pressure is calculated for steps between energy outputs. Berendsen method uses average rather then instantaneous pressure. Ratio of first two basis vectors can be fixed for flexible cells. Any contiguous fragment can be wrapped to the periodic cell on output, rather than only wrapping water molecules. Coordinates can be wrapped to the true nearest image for hexagonal or similar shaped periodic cells.

Additional Parameters Can Be Changed During a Simulation Script

Fixed atoms can be turned off during a run. Most pressure and temperature control parameters can now be changed as well.

Fixes to AMBER parm File Reader for AMBER 7 or Large Molecules

Fixed problems reading parm files containing more than 33,333 atoms. Also added changes to read new format parm files from AMBER 7.

1.2 NAMD and molecular dynamics simulations

Molecular dynamics (MD) simulations compute atomic trajectories by solving equations of motion numerically using empirical force fields, such as the CHARMM force field, that approximate the actual atomic force in biopolymer systems. Detailed information about MD simulations can be found in several books such as [1, 14]. In order to conduct MD simulations, various computer programs have been developed including X-PLOR [7] and CHARMM [6]. These programs were originally developed for serial machines. Simulation of large molecules, however, require enormous computing power. One way to achieve such simulations is to utilize parallel computers. In recent years, distributed memory parallel computers have been offering cost-effective computational power. NAMD was designed to run efficiently on such parallel machines for simulating large molecules. NAMD is particularly well suited to the increasingly popular Beowulf-class PC clusters, which are quite similar to the workstation clusters for which is was originally designed. Future versions of NAMD will also make efficient use of clusters of multi-processor workstations or PCs.

NAMD has several important features:

- **Force Field Compatibility**
  The force field used by NAMD is the same as that used by the programs CHARMM [6] and X-PLOR [7]. This force field includes local interaction terms consisting of bonded interactions between 2, 3, and 4 atoms and pairwise interactions including electrostatic and van der Waals forces. This commonality allows simulations to migrate between these three programs.

- **Efficient Full Electrostatics Algorithms**
  NAMD incorporates the Particle Mesh Ewald (PME) algorithm, which takes the full electrostatic interactions into account. This algorithm reduces the computational complexity of electrostatic force evaluation from $O(N^2)$ to $O(N \log N)$.

- **Multiple Time Stepping**
  The velocity Verlet integration method [1] is used to advance the positions and velocities of the atoms in time. To further reduce the cost of the evaluation of long-range electrostatic forces, a multiple time step scheme is employed. The local interactions (bonded, van der Waals and electrostatic interactions within a specified distance) are calculated at each time step. The longer range interactions (electrostatic interactions beyond the specified distance) are only computed less often. This amortizes the cost of computing the electrostatic forces over several timesteps. A smooth splitting function is used to separate a quickly varying short-range portion of the electrostatic interaction from a more slowly varying long-range component. It is also possible to employ an intermediate timestep for the short-range non-bonded interactions, performing only bonded interactions every timestep.

- **Input and Output Compatibility**
  The input and output file formats used by NAMD are identical to those used by CHARMM and X-PLOR. Input formats include coordinate files in PDB format [3], structure files in X-PLOR PSF format, and energy parameter files in either CHARMM or X-PLOR formats. Output formats include PDB coordinate files and binary DCD trajectory files. These similarities assure that the molecular dynamics trajectories from NAMD can be read by CHARMM or X-PLOR and that the user can exploit the many analysis algorithms of the latter packages.

- **Dynamics Simulation Options**
  MD simulations may be carried out using several options, including

  - Constant energy dynamics,
  - Constant temperature dynamics via
    * Velocity rescaling,
    * Velocity reassignment,
    * Langevin dynamics,
  - Periodic boundary conditions,
  - Constant pressure dynamics via
    * Berendsen pressure coupling,
    * Nosé-Hoover Langevin piston,
  - Energy minimization,
  - Fixed atoms,
  - Rigid waters,
  - Rigid bonds to hydrogen,
  - Harmonic restraints,
  - Spherical or cylindrical boundary restraints.

- **Easy to Modify and Extend**
  Another primary design objective for NAMD is extensibility and maintainability. In order to achieve this, it is designed in an object-oriented style with C++. Since molecular dynamics is a new field, new algorithms and techniques are continually being developed. NAMD's modular design allows one to integrate and test new algorithms easily. If you are contemplating a particular modification to NAMD you are encouraged to contact the developers at `namd@ks.uiuc.edu` for guidance.

- **Interactive MD simulations**
  A system undergoing simulation in NAMD may be viewed and altered with VMD; for instance, forces can be applied to a set of atoms to alter or rearrange part of the molecular structure. For more information on VMD, see `http://www.ks.uiuc.edu/Research/vmd/`.

- **Load Balancing**
  An important factor in parallel applications is the equal distribution of computational load among the processors. In parallel molecular simulation, a spatial decomposition that evenly distributes the computational load causes the region of space mapped to each processor to become very irregular, hard to compute and difficult to generalize to the evaluation of many different types of forces. NAMD addresses this problem by using a simple uniform spatial decomposition where the entire model is split into uniform cubes of space called *patches*. An initial load balancer assigns patches and the calculation of interactions among the atoms within them to processors such that the computational load is balanced as much as possible. During the simulation, an incremental load balancer monitors the load and performs necessary adjustments.

1.3 User feedback

If you have problems installing or running NAMD after reading this document, please send a complete description of the problem by email to `namd@ks.uiuc.edu`. If you discover and fix a problem not described in this manual we would appreciate if you would tell us about this as well, so we can alert other users and incorporate the fix into the public distribution.

We are interested in making NAMD more useful to the molecular modeling community. Your suggestions are welcome at `namd@ks.uiuc.edu`. We also appreciate hearing about how you are using NAMD in your work.

1.4 Acknowledgments

This work is supported by grants from the National Science Foundation (BIR-9318159) and the National Institute of Health (PHS 5 P41 RR05969-04).

The authors would particularly like to thank the members of the Theoretical Biophysics Group, past and present, who have helped tremendously in making suggestions, pushing for new features, and testing bug-ridden code.

2 Getting Started

2.1 What is needed

Before running NAMD, explained in section 9, the following are be needed:

- A CHARMM force field in either CHARMM or X-PLOR format.
- An X-PLOR format PSF file describing the molecular structure.
- The initial coordinates of the molecular system in the form of a PDB file.
- A NAMD configuration file.

NAMD provides the `psfgen` utility, documented in Section 4, which is capable of generating the required PSF and PDB files by merging PDB files and guessing coordinates for missing atoms. If `psfgen` is insufficient for your system, we recommend that you obtain access to either CHARMM or X-PLOR, both of which are capable of generating the required files.

2.2 NAMD configuration file

Besides these input and output files, NAMD also uses a file referred to as the *configuration file*. This file specifies what dynamics options and values that NAMD should use, such as the number of timesteps to perform, initial temperature, etc. The options and values in this file control how the system will be simulated.

A NAMD configuration file contains a set of options and values. The options and values specified determine the exact behavior of NAMD, what features are active or inactive, how long the simulation should continue, etc. Section 2.2.1 describes how options are specified within a NAMD configuration file. Section 2.2.3 lists the parameters which are required to run a basic simulation. Section 7 describes the relation between specific NAMD and X-PLOR dynamics options. Several sample NAMD configuration files are shown in section 8.

2.2.1 Configuration parameter syntax

Each line in the configuration files consists of a *keyword* identifying the option being specified, and a *value* which is a parameter to be used for this option. The keyword and value can be separated by only white space:

```
keyword             value
```

or the keyword and value can be separated by an equal sign and white space:

```
keyword      =      value
```

Blank lines in the configuration file are ignored. Comments are prefaced by a # and may appear on the end of a line with actual values:

```
keyword             value         #  This is a comment
```

or may be at the beginning of a line:

```
#  This entire line is a comment . . .
```

Some keywords require several lines of data. These are generally implemented to either allow the data to be read from a file:

```
keyword             filename
```

or to be included inline using Tcl-style braces:

```
keyword {
  lots of data
}
```

The specification of the keywords is case insensitive so that any combination of upper and lower case letters will have the same meaning. Hence, `DCDfile` and `dcdfile` are equivalent. The capitalization in the values, however, may be important. Some values indicate file names, in which capitalization is critical. Other values such as `on` or `off` are case insensitive.

2.2.2 Tcl scripting interface and features

When compiled with Tcl (all released binaries) the config file is parsed by Tcl in a fully backwards compatible manner with the added bonus that any Tcl command may also be used. This alone allows:

- the ”source” command to include other files (works w/o Tcl too!),
- the ”print” command to display messages (”puts” is broken, sorry),
- environment variables through the env array (”$env(USER)”), and
- user-defined variables (”set base sim23”, ”dcdfile $base.dcd”).

Additional features include:

- The ”callback” command takes a 2-parameter Tcl procedure which is then called with a list of labels and a list of values during every timestep, allowing analysis, formatting, whatever.
- The ”run” command takes a number of steps to run (overriding the now optional numsteps parameter, which defaults to 0) and can be called repeatedly. You can ”run 0” just to get energies.
- The ”minimize” command is similar to ”run” and performs minimization for the specified number of force evaluations.
- The ”output” command takes an output file basename and causes .coor, .vel, and .xsc files to be written with that name.
- Between ”run” commands the reassignTemp, rescaleTemp, and langevinTemp parameters can be changed to allow simulated annealing protocols within a single config file. The useGroupPressure, useFlexibleCell, useConstantArea, useConstantRatio, LangevinPiston, LangevinPistonTarget, LangevinPistonPeriod, LangevinPistonDecay, LangevinPistonTemp, SurfaceTensionTarget, BerendsenPressure, BerendsenPressureTarget, BerendsenPressureCompressibility, and BerendsenPressureRelaxationTime parameters may be changed to allow pressure equilibration. The fixedAtoms and constraintScaling parameters may be changed to preserve macromolecular conformation during minimization and equilibration (fixedAtoms may only be disabled, and requires that fixedAtomsForces is enabled to do this).

- The "checkpoint" and "revert" commands (no arguments) allow a scripted simulation to save and restore to a prior state.
- The "reinitvels" command reinitializes velocities to a random distribution based on the given temperature.
- The "measure" command allows user-programmed calculations to be executed in order to facilitate automated methods. (For example, to revert or change a parameter.) You will need to write code and compile NAMD to make use of this feature.

Please note that while NAMD has traditionally allowed comments to be started by a # appearing anywhere on a line, Tcl only allows comments to appear where a new statement could begin. With Tcl config file parsing enabled (all shipped binaries) both NAMD and Tcl comments are allowed before the first "run" command. At this point only pure Tcl syntax is allowed. In addition, the ";#" idiom for Tcl comments will only work with Tcl enabled. NAMD has also traditionally allowed parameters to be specified as "param=value". This is supported, but only before the first "run" command. Some examples:

```
# this is my config file                        <- OK
reassignFreq 100 ; # how often to reset velocities  <- only w/ Tcl
reassignTemp 20 # temp to reset velocities to      <- OK before "run"
run 1000                                           <- now Tcl only
reassignTemp 40 ; # temp to reset velocities to    <- ";" is required
```

NAMD has also traditionally allowed parameters to be specified as "param=value" as well as "param value". This is supported, but only before the first "run" command. For an easy life, use "param value".

2.2.3 Required NAMD configuration parameters

The following parameters are *required* for every NAMD simulation:

- `numsteps` (page 39),
- `coordinates` (page 18),
- `structure` (page 18),
- `parameters` (page 18),
- `exclude` (page 40),
- `outputname` (page 19),
- one of the following three:
  - `temperature` (page 41),
  - `velocities` (page 19),
  - `binvelocities` (page 19).

These required parameters specify the most basic properties of the simulation. In addition, it is highly recommended that `pairlistdist` be specified with a value at least one greater than `cutoff`.

3 Input and Output Files

NAMD was developed to be compatible with existing molecular dynamics packages, especially the packages X-PLOR [7] and CHARMM [6]. To achieve this compatibility, the set of input files which NAMD uses to define a molecular system are identical to the input files used by X-PLOR and CHARMM. Thus it is trivial to move an existing simulation from X-PLOR or CHARMM to NAMD. A description of these molecular system definition files is given in Section 3.1.

In addition, the output file formats used by NAMD were chosen to be compatible with X-PLOR and CHARMM. In this way the output from NAMD can be analyzed using X-PLOR, CHARMM, or a variety of the other tools that have been developed for the existing output file formats. Descriptions of the output files formats are also given in Section 3.1.

3.1 File formats

3.1.1 PDB files

The PDB (Protein Data Bank) format is used to store coordinate or velocity data being input or output from NAMD. This is the standard format for coordinate data for most other molecular dynamics programs as well, including X-PLOR and CHARMM. A full description of this file format can be obtained via anonymous FTP from `ftp.pdb.bnl.gov` in `/pub/format.desc.ps.Z` or `/pub/format.desc.txt`.

3.1.2 X-PLOR format PSF files

NAMD uses the same protein structure files that X-PLOR does. At this time, the easiest way to generate these files is using X-PLOR or CHARMM, although it is possible to build them by hand. CHARMM can generate an X-PLOR format PSF file with the command "`write psf card xplor`".

3.1.3 CHARMM19 and CHARMM22 parameter files

NAMD supports CHARMM19 and CHARMM22 parameter files in both X-PLOR and CHARMM formats. (X-PLOR format is the default, CHARMM format parameter files may be used given the parameter "`paraTypeCharmm on`".) For a full description of the format of commands used in these files, see the X-PLOR and CHARMM User's Manual [7].

3.1.4 DCD trajectory files

NAMD produces DCD trajectory files in the same format as X-PLOR and CHARMM. The DCD files are single precision binary FORTRAN files, so are transportable between computer architectures. They are not, unfortunately, transportable between big-endian (most workstations) and little endian (Intel) architectures. (This same caveat applies to binary velocity and coordinate files. The utility programs `flipdcd` and `flipbinpdb` are provided with the Linux/Intel version to reformat these files.) The exact format of these files is very ugly but supported by a wide range of analysis and display programs.

3.2 NAMD configuration parameters

Input files

- `coordinates` < coordinate PDB file >
  Acceptable Values: UNIX filename
  Description: The PDB file containing initial position coordinate data. Note that path names can be either absolute or relative. Only one value may be specified.

- `structure` < PSF file >
  Acceptable Values: UNIX filename
  Description: The X-PLOR format PSF file describing the molecular system to be simulated. Only one value may be specified.

- `parameters` < parameter file >
  Acceptable Values: UNIX filename
  Description: A CHARMM19 or CHARMM22 parameter file that defines all or part of the parameters necessary for the molecular system to be simulated. At least one parameter file must be specified for each simulation. Multiple definitions are allowed for systems that require more than one parameter file. For example, if three files were needed, lines such as:

  ```
  parameters param1

  parameters param2

  parameters param3
  ```

  could be added to the configuration file. The files will be read in the order that they appear in the configuration file. If duplicate parameters are read, a warning message is printed and the last parameter value read is used. Thus, the order that files are read can be important in cases where duplicate values appear in separate files.

- `paraTypeXplor` < Is the parameter file in X-PLOR format? >
  Acceptable Values: `on` or `off`
  Default Value: `on`
  Description: Specifies whether or not the parameter file(s) are in X-PLOR format. X-PLOR format is the default for parameter files! Caveat: The PSF file should be also constructed with X-PLOR in case of an X-PLOR parameter file because X-PLOR stores information about the multiplicity of dihedrals in the PSF file. See the X-PLOR manual for details.

- `paraTypeCharmm` < Is the parameter file in CHARMM format? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not the parameter file(s) are in CHARMM format. X-PLOR format is the default for parameter files! Caveat: The information about multiplicity of dihedrals will be obtained directly from the parameter file, and the full multiplicity will be used (same behavior as in CHARMM). If the PSF file originates from X-PLOR, consecutive multiple entries for the same dihedral (indicating the dihedral multiplicity for X-PLOR) will be ignored.

- `velocities` < velocity PDB file >
  Acceptable Values: UNIX filename

Description: The PDB file containing the initial velocities for all atoms in the simulation. This is typically a restart file or final velocity file written by NAMD during a previous simulation. Either the `temperature` or the `velocities/binvelocities` option must be defined to determine an initial set of velocities. Both options cannot be used together.

- `binvelocities` < binary velocity file >
  Acceptable Values: UNIX filename
  Description: The binary file containing initial velocities for all atoms in the simulation. A binary velocity file is created as output from NAMD by activating the `binaryrestart` or `binaryoutput` options. The `binvelocities` option should be used as an alternative to `velocities`. Either the `temperature` or the `velocities/binvelocities` option must be defined to determine an initial set of velocities. Both options cannot be used together.

- `bincoordinates` < binary coordinate restart file >
  Acceptable Values: UNIX filename
  Description: The binary restart file containing initial position coordinate data. A binary coordinate restart file is created as output from NAMD by activating the `binaryrestart` or `binaryoutput` options. Note that, in the current implementation at least, the `bincoordinates` option must be used in addition to the `coordinates` option, but the positions specified by `coordinates` will then be ignored.

- `cwd` < default directory >
  Acceptable Values: UNIX directory name
  Description: The default directory for input and output files. If a value is given, all filenames that do not begin with a / are assumed to be in this directory. For example, if `cwd` is set to `/scr`, then a filename of `outfile` would be modified to `/scr/outfile` while a filename of `/tmp/outfile` would remain unchanged. If no value for `cwd` is specified, than all filenames are left unchanged *but are assumed to be relative to the directory which contains the configuration file given on the command line.*

Output files

- `outputname` < output PDB file >
  Acceptable Values: UNIX filename prefix
  Description: At the end of every simulation, NAMD writes two PDB files, one containing the final coordinates and another containing the final velocities of all atoms in the simulation. This option specifies the file prefix for these two files. The position coordinates will be saved to a file named as this prefix with `.coor` appended. The velocities will be saved to a file named as this prefix with `.vel` appended. For example, if the prefix specified using this option was `/tmp/output`, then the two files would be `/tmp/output.coor` and `/tmp/output.vel`.

- `binaryoutput` < use binary output files? >
  Acceptable Values: `yes` or `no`
  Default Value: `yes`
  Description: Activates the use of binary output files. If this option is set to `yes`, then the final output files will be written in binary rather than PDB format. Binary files preserve more accuracy between NAMD restarts than ASCII PDB files, but the binary files are not guaranteed to be transportable between computer architectures. (The utility program `flipbinpdb` is provided with the Linux/Intel version to reformat these files.)

- `restartname` < restart files >
  Acceptable Values: UNIX filename prefix
  Description: The prefix to use for restart filenames. NAMD produces PDB restart files that store the current positions and velocities of all atoms at some step of the simulation. This option specifies the prefix to use for restart files in the same way that `outputname` specifies a filename prefix for the final positions and velocities. If `restartname` is defined then the parameter `restartfreq` must also be defined.

- `restartfreq` < frequency of restart file generation >
  Acceptable Values: positive integer
  Description: The number of timesteps between the generation of restart files. If `restartfreq` is defined, then `restartname` must also be defined.

- `restartsave` < use timestep in restart filenames? >
  Acceptable Values: `yes` or `no`
  Default Value: `no`
  Description: Appends the current timestep to the restart filename prefix, producing a sequence of restart files rather than only the last version written.

- `binaryrestart` < use binary restart files? >
  Acceptable Values: `yes` or `no`
  Default Value: `yes`
  Description: Activates the use of binary restart files. If this option is set to `yes`, then the restart files will be written in binary rather than PDB format. Binary files preserve more accuracy between NAMD restarts than ASCII PDB files, but the binary files are not guaranteed to be transportable between computer architectures. (The utility program `flipbinpdb` is provided with the Linux/Intel version to reformat these files.)

- `DCDfile` < coordinate trajectory output file >
  Acceptable Values: UNIX filename
  Description: The binary DCD position coordinate trajectory filename. This file stores the trajectory of all atom position coordinates using the same format (binary DCD) as X-PLOR. If `DCDfile` is defined, then `DCDfreq` must also be defined.

- `DCDfreq` < timesteps between writing coordinates to trajectory file >
  Acceptable Values: positive integer
  Description: The number of timesteps between the writing of position coordinates to the trajectory file. The initial positions will not be included in the trajectory file.

- `velDCDfile` < velocity trajectory output file >
  Acceptable Values: UNIX filename
  Description: The binary DCD velocity trajectory filename. This file stores the trajectory of all atom velocities using the same format (binary DCD) as X-PLOR. If `velDCDfile` is defined, then `velDCDfreq` must also be defined.

- `velDCDfreq` < timesteps between writing velocities to trajectory file >
  Acceptable Values: positive integer
  Description: The number of timesteps between the writing of velocities to the trajectory file. The initial velocities will not be included in the trajectory file.

- `outputEnergies` < timesteps between energy output >
  Acceptable Values: positive integer
  Default Value: 1
  Description: The number of timesteps between each energy output of NAMD. This value specifies how often NAMD should output the current energy values to **stdout** (which can be redirected to a file). By default, this is done every step. For long simulations, the amount of output generated by NAMD can be greatly reduced by outputting the energies only occasionally.

- `outputMomenta` < timesteps between momentum output >
  Acceptable Values: nonnegative integer
  Default Value: 0
  Description: The number of timesteps between each momentum output of NAMD. If specified and nonzero, linear and angular momenta will be output to **stdout**.

- `outputPressure` < timesteps between pressure output >
  Acceptable Values: nonnegative integer
  Default Value: 0
  Description: The number of timesteps between each pressure output of NAMD. If specified and nonzero, atomic and group pressure tensors will be output to **stdout**.

- `outputTiming` < timesteps between timing output >
  Acceptable Values: nonnegative integer
  Default Value: 0
  Description: The number of timesteps between each timing output of NAMD. If specified and nonzero, CPU and wallclock times will be output to **stdout**. These data are from node 0 only; CPU times for other nodes may vary.

3.3 AMBER force field parameters

AMBER format PARM file and coordinate file can be read by NAMD, which allows one to use AMBER force field to carry out all types of simulations that NAMD has supported. NAMD can read PARM files in either the format used in AMBER 6 or the new format defined in AMBER 7. The output of the simulation (restart file, DCD file, etc.) will still be in traditional format that has been used in NAMD.

- `amber` < use AMBER format force field? >
  Acceptable Values: yes or no
  Default Value: no
  Description: If `amber` is set to on, then `parmfile` must be defined, and `structure` and `parameters` should not be defined.

- `parmfile` < AMBER format PARM file >
  Acceptable Values: UNIX filename
  Description: This file contains complete topology and parameter information of the system.

- `ambercoor` < AMBER format coordinate file >
  Acceptable Values: UNIX filename

Description: This file contains the coordinates of all the atoms. Note that `coordinates` can also be used for PDB format coordinate file. When `amber` is set to on, either `ambercoor` or `coordinates` must be defined, but not both.

- `readexclusions` < Read exclusions from PARM file? >
  Acceptable Values: yes or no
  Default Value: yes
  Description: PARM file explicitly gives complete exclusion (including 1-4 exclusions) information. When `readexclusions` is set to on, NAMD will read all exclusions from PARM file and will not add any more; alternatively, if `readexclusions` is set to off, NAMD will ignore the exclusions in PARM file and will automatically generate them according to the exclusion policy specified by `exclude`.

- `scnb` < VDW 1-4 scaling factor >
  Acceptable Values: decimal ≥ 1.0
  Default Value: 2.0
  Description: Same meaning as SCNB in AMBER. Note that in NAMD, 1-4 vdw interactions are DIVIDED by `scnb`, whereas 1-4 electrostatic interactions are MULTIPLIED by `1-4scaling`. So `1-4scaling` should be set to the inverse of SCEE value used in AMBER.

Caveat:
1. Polarizable parameters in AMBER are not supported.
2. NAMD does not support the 10-12 potential terms in some old AMBER versions. When non-zero 10-12 parameter is encountered in PARM file, NAMD will terminate.
3. NAMD has several exclusion policy options, defined by `exclude`. The way AMBER dealing with exclusions corresponds to the "scaled1-4" in NAMD. So for simulations using AMBER force field, one would specify "exclude scaled1-4" in the configuration file, and set `1-4scaling` to the inverse value of SCEE as would be used in AMBER.
4. NAMD does not read periodic box lengths in PARM or coordinate file. They must be explicitly specified in NAMD configuration file.
5. By default, NAMD applies switching functions to the non-bond interactions within the cut-off distance, which helps to improve energy conservation, while AMBER does not use switching functions so it simply truncates the interactions at cutoff. However, if "authentic" AMBER cutoff simulations are desired, the switching functions could be turned off by specifying "switching off" in NAMD configuration file.
6. When SHAKE is applied to water molecules, NAMD constrains the two O-H bonds and the H-O-H angle, which produces the same result as constraining the additional H-H bond. But in some AMBER models, water is defined by three real bonds and no angle. In this case NAMD will apply SHAKE only to the two O-H bonds and not to the angle or the H-H bond (because the angle is not defined). Thus, when SHAKE is used along with this kind of water model, difference would be expected between NAMD and AMBER.
7. NAMD and AMBER may have different default values for some parameters (e.g., the tolerance of SHAKE). One should check other sections of this manual for accurate descriptions of the NAMD options.

Following are two examples of the NAMD configuration file to read AMBER force field and carry out simulation. They may help users to select proper NAMD options for AMBER force field. For the convenience of AMBER users, the AMBER 6 sander input files are given in the left for comparison, which would accomplish similar tasks in AMBER.

Example 1: Non-periodic boundary system, cutoff simulation

```
---AMBER----       ---NAMD---

 TITLE
 &cntrl
  ntb=0, igb=2,    # non-periodic, use cutoff for non-bond
  nstlim=1000,     numsteps       1000  # Num of total steps
  ntpr=50,         outputEnergies 50  # Energy output frequency
  ntwr=50,         restartfreq    50  # Restart file frequency
  ntwx=100,        DCDfreq        100  # Trajectory file frequency
  dt=0.001,        timestep       1  # in unit of fs (This is default)
  tempi=0.,        temperature    0  # Initial temp for velocity assignment
  cut=10.,         cutoff         10
                   switching      off  # Turn off the switching functions
  scee=1.2,        exclude        scaled1-4
                   1-4scaling     0.833333  # =1/1.2, default is 1.0
  scnb=2.0         scnb           2  # This is default
 &end
                   amber          on  # Specify this is AMBER force field
                   parmfile       prmtop  # Input PARM file
                   ambercoor      inpcrd  # Input coordinate file
                   outputname     md  # Prefix of output files
```

Example 2: Periodic boundary system, PME, NVE ensemble, using SHAKE algorithm

```
---AMBER----       ---NAMD---

 TITLE
 &cntrl
  ntc=2, ntf=2,    # SHAKE to the bond between each hydrogen and it mother atom
                   rigidBonds     all
  tol=0.0005,      rigidTolerance 0.0005  # Default is  0.00001
  nstlim=500,      numsteps       500  # Num of total steps
  ntpr=50,         outputEnergies 50  # Energy output frequency
  ntwr=100,        restartfreq    100  # Restart file frequency
  ntwx=100,        DCDfreq        100  # Trajectory file frequency
  dt=0.001,        timestep       1  # in unit of fs (This is default)
  tempi=300.,      temperature    300  # Initial temp for velocity assignment
  cut=9.,          cutoff         9
                   switching      off  # Turn off the switching functions
 &end
 &ewald            PME            on  # Use PME for electrostatic calculation
                   # Orthogonal periodic box size
  a=62.23,         cellBasisVector1   62.23  0  0
  b=62.23,        cellBasisVector2   0  62.23  0
  c=62.23,        cellBasisVector3   0  0  62.23
  nfft1=64,       PMEGridSizeX   64
  nfft2=64,       PMEGridSizeY   64
  nfft3=64,       PMEGridSizeZ   64
  ischrgd=1,      # NAMD doesn't force neutralization of charge
 &end
                  amber          on  # Specify this is AMBER force field
                  parmfile       FILENAME  # Input PARM file
                  ambercoor      FILENAME  # Input coordinate file
                  outputname     PREFIX  # Prefix of output files
                  exclude        scaled1-4
                  1-4scaling     0.833333  # =1/1.2, default is 1.0
```

3.4 GROMACS force field paramets

NAMD has the ability to load GROMACS ASCII topology (.top) and coordinate (.gro) files, which allows you to run most GROMACS simulations in NAMD. All simulation output will still be in the traditional NAMD formats.

- `gromacs` < use GROMACS format force field? >
  Acceptable Values: on or off
  Default Value: off
  Description: If `gromacs` is set to on, then `grotopfile` must be defined, and `structure` and `parameters` should not be defined.

- `grotopfile` < GROMACS format topology/parameter file >
  Acceptable Values: UNIX filename
  Description: This file contains complete topology and parameter information of the system.

- `grocoorfile` < GROMACS format coordinate file >
  Acceptable Values: UNIX filename
  Description: This file contains the coordinates of all the atoms. Note that `coordinates` can also be used for PDB format coordinate file. When `gromacs` is set to `on`, either `grocoorfile` or `coordinates` must be defined, but not both.

However, NAMD does not have support for many GROMACS-specific options:

- Dummies (fake atoms with positions generated from the positions of real atoms) are not supported.

- The GROMACS `pairs` section, where explicit 1–4 parameters are given between pairs of atoms, is not supported, since NAMD calculates its 1–4 interactions exclusively by type.

- Similarly, `exclusions` are not supported. The biggest problem here is that GROMACS RB dihedrals are supposed to imply exclusions, but NAMD does not support this.

- Constraints, restraints, and `settles` are not implemented in NAMD.

- In some cases, it may not work to override some but not all of the parameters for a bond, atom, etc. In this case, NAMD will generate an error and stop. The parser will sometimes not tolerate correct GROMACS files or fail to detect errors in badly formatted files.
- NAMD does not support all the types of bond potentials that exist in GROMACS, but approximates them with harmonic or sinusoidal potentials.
- NAMD does not read periodic box lengths in the coordinate file. They must be explicitly specified in the NAMD configuration file.

4 Creating PSF Structure Files

The `psfgen` structure building tool consists of a portable library of structure and file manipulation routines with a Tcl interface. Current capabilities include

- reading CHARMM topology files
- reading psf files in X-PLOR/NAMD format
- extracting sequence data from single segment PDB files
- generating a full molecular structure from sequence data
- applying patches to modify or link different segments
- writing NAMD and VMD compatible PSF structure files
- extracting coordinate data from PDB files
- constructing (guessing) missing atomic coordinates
- deleting selected atoms from the structure
- writing NAMD and VMD compatible PDB coordinate files

We are currently refining the interface of `psfgen` and adding features to create a complete molecular building solution. We welcome your feedback on this new tool.

4.1 Ordinary Usage

`psfgen` is currently distributed in two forms. One form is as a standalone program implemented as a Tcl interpreter which reads commands from standard output. You may use loops, variables, etc. as you would in a VMD or NAMD script. You may use psfgen interactively, but we expect it to be run most often with a script file redirected to standard input. The second form is as a Tcl package which can be imported into any Tcl application, including VMD. All the commands available to the standalone version of psfgen are available to the Tcl package; using `psfgen` within VMD lets you harness VMD's powerful atom selection capability, as well as instantly view the result of your structure building scripts. Examples of using `psfgen` both with and without VMD are provided in this document.

Generating PSF and PDB files for use with NAMD will typically consist of the following steps:

1. Preparing separate PDB files containing individual segments of protein, solvent, etc. before running `psfgen`.
2. Reading in the appropriate topology definition files and aliasing residue and atom names found in the PDB file to those found in the topology files. This will generally include selecting a default protonation state for histidine residues.
3. Generating the default structure using segment and pdb commands.
4. Applying additional patches to the structure.
5. Reading coordinates from the PDB files.

6. Deleting unwanted atoms, such as overlapping water molecules.
7. Guessing missing coordinates of hydrogens and other atoms.
8. Writing PSF and PDB files for use in NAMD.

4.2 List of Commands

- `topology` <*file name*>
  Purpose: Read in molecular topology definitions from file.
  Arguments: <*file name*>: CHARMM format topology file.
  Context: Beginning of script, before segment. May call multiple times.

- `alias residue` <*alternate name*> <*real name*>
  Purpose: Provide translations from residues found in PDB files to proper residue names read in from topology definition files. Proper names from topology files will be used in generated PSF and PDB files.
  Arguments: <*alternate name*>: Residue name found in PDB file.
  <*real name*>: Residue name found in topology file.
  Context: Before reading sequence with pdb. May call multiple times.

- `segment` <*segment ID*> { <*commands*> }
  Purpose: Build a segment of the molecule. A segment is typically a single chain of protein or DNA, with default patches applied to the termini. Segments may also contain pure solvent or lipid.
  Arguments: <*segment ID*>: Unique name for segment, 1–4 characters.
  <*commands*>: Sequence of commands in Tcl syntax to build the primary structure of the segment, including auto, first, last, residue, pdb, etc.
  Context: After topology definitions and residue aliases. May call multiple times. Structure information is generated at the end of every segment command.

- `auto` [`angles`] [`dihedrals`] [`none`]
  Purpose: Override default settings from topology file for automatic generation of angles and dihedrals for the current segment.
  Arguments: `angles`: Enable generation of angles from bonds.
  `dihedrals`: Enable generation of dihedrals from angles.
  `none`: Disable generation of angles and dihedrals.
  Context: Anywhere within segment, does not affect later segments.

- `first` <*patch name*>
  Purpose: Override default patch applied to first residue in segment. Default is read from topology file and may be residue-specific.
  Arguments: <*patch name*>: Single-target patch residue name or `none`.
  Context: Anywhere within segment, does not affect later segments.

- `last` <*patch name*>
  Purpose: Override default patch applied to last residue in segment. Default is read from topology file and may be residue-specific.
  Arguments: <*patch name*>: Single-target patch residue name or `none`.
  Context: Anywhere within segment, does not affect later segments.

- `residue` <*resid*> <*resname*>
  Purpose: Add a single residue to the end of the current segment.
  Arguments: <*resid*>: Unique name for residue, 1–5 characters, usually numeric. <*resname*>: Residue type name from topology file.
  Context: Anywhere within segment.

- `pdb` <*file name*>
  Purpose: Extract sequence information from PDB file when building segment. Residue IDs will be preserved, residue names must match entries in the topology file or should be aliased before pdb is called.
  Arguments: <*file name*>: PDB file containing known or aliased residues.
  Context: Anywhere within segment.

- `mutate` <*resid*> <*resname*>
  Purpose: Change the type of a single residue in the current segment.
  Arguments: <*resid*>: Unique name for residue, 1–5 characters, usually numeric. <*resname*>: New residue type name from topology file.
  Context: Within segment, after target residue has been created.

- `patch` <*patch residue name*> <*segid:resid*> [*...*]
  Purpose: Apply a patch to one or more residues. Patches make small modifications to the structure of residues such as converting one to a terminus, changing the protonation state, or creating disulphide bonds between a pair of residues.
  Arguments: <*patch residue name*>: Name of patch residue from topology definition file. <*segid:resid*>: List of segment and residue pairs to which patch should be applied.
  Context: After one or more segments have been built.

- `multiply` <*factor*> <*segid*[*:resid*[*:atomname*]]> [*...*]
  Purpose: Create multiple images of a set of atoms for use in locally enhanced sampling. The beta column of the output pdb file is set to 1...<*factor*> for each image. Multiple copies of bonds, angles, etc. are created. Atom, residue or segment names are not altered; images are distinguished only by beta value. This is not a normal molecular structure and may confuse other tools.
  Arguments: <*factor*>:
  <*segid:resid:atomname*>: segment, residue, or atom to be multiplied. If :resid is omitted the entire segment is multiplied; if :atomname is omitted the entire residue is multiplied. May be repeated as many times as necessary to include all atoms.
  Context: After one or more segments have been built, all patches applied, and coordinates guessed. The effects of this command may confuse other commands.

- `delatom` <*segid*> [*resid*] [*atom name*]
  Purpose: Delete one or more atoms. If only `segid` is specified, all atoms from that segment will be removed from the structure. If both `segid` and `resid` are specified, all atoms from just that residue will be removed. If `segid`, `resid`, and `atom name` are all specified, just a single atom will be removed.
  Arguments: <*segid*>: Name of segment.
  <*resid*>: Name of residue (optional).
  <*atom name*>: Name of atom (optional).
  Context: After all segments have been built and patched.

- `resetpsf`
  Purpose: Delete all segments in from the structure. The topology definitions and aliases are left intact.
  Arguments:
  Context: After one or more segments have been built.

- `writepsf` [`charmm`] [`x-plor`] <*file name*>
  Purpose: Write out structure information as PSF file.
  Arguments: `charmm`: Use CHARMM format (numbers for atom types).
  `x-plor`: Use X-PLOR format (names for atom types), the default format required by NAMD.
  <*file name*>: PSF file to be generated.
  Context: After all segments have been built and patched.

- `readpsf` <*file name*>
  Purpose: Read in structure information from PSF file and adds it to the structure. It is an error if any segments in the PSF file already exist.
  Arguments: <*file name*>: PSF file in X-PLOR format (names for atom types).
  Context: Anywhere but within segment.

- `alias atom` <*residue name*> <*alternate name*> <*real name*>
  Purpose: Provide translations from atom names found in PDB files to proper atom names read in from topology definition files. Proper names from topology files will be used in generated PSF and PDB files.
  Arguments: <*residue name*>: Proper or aliased residue name.
  <*alternate name*>: Atom name found in PDB file.
  <*real name*>: Atom name found in topology file.
  Context: Before reading coordinates with coordpdb. May call multiple times.

- `coord` <*segid*> <*resid*> <*atomname*> <{ $x\ y\ z$ }>
  Purpose: Set coordinates for a single atom.
  Arguments: <*segid*>: Segment ID of target atom.
  <*resid*>: Residue ID of target atom.
  <*atomname*>: Name of target atom.
  <{ $x\ y\ z$ }>: Coordinates to be assigned.
  Context: After structure has been generated.

- `coordpdb` <*file name*> [*segid*]
  Purpose: Read coordinates from PDB file, matching segment, residue and atom names.
  Arguments: <*file name*>: PDB file containing known or aliased residues and atoms.
  <*segid*>: If specified override segment IDs in PDB file.
  Context: After segment has been generated and atom aliases defined.

- `guesscoord`
  Purpose: Guesses coordinates of atoms for which they were not explicitly set. Calculation is based on internal coordinate hints contained in toplogy definition files. When these are insufficient, wild guesses are attempted based on bond lengths of 1 Å and angles of 109°.
  Arguments: None.
  Context: After stucture has been generated and known coordinates read in.

- `writepdb` <*file name*>
  Purpose: Writes PDB file containing coordinates. Atoms order is identical to PSF file generated by writepsf (unless structure has been changed). The O field is set to 1 for atoms with known coordinates, 0 for atoms with guessed coordinates, and -1 for atoms with no coordinate data available (coordinates are set to 0 for these atoms).
  Arguments: <*file name*>: PDB file to be written.
  Context: After structure and coordinates are complete.

4.3 BPTI Example

To actually run this demo requires

- the program `psfgen` from any NAMD distribution,
- the CHARMM topology and parameter files `top_all22_prot.inp` and `par_all22_prot.inp` from `https://rxsecure.umaryland.edu/research/amackere/research.html`, and
- the BPTI PDB file `6PTI.pdb` available from the Protein Data Bank at `http://www.pdb.org/` by searching for `6PTI` and downloading the complete structure file in PDB format.

In this demo, we create the files `bpti.psf` and `bpti.pdb` in the output directory which can then be used for a simple NAMD simulation.

Create the working directory. Nothing outside of the directory `output` is modified.

```
mkdir output
```

Splitting input PDB file into segments. 6PTI.pdb is the original file from the Protein Data Bank. It contains a single chain of protein and some PO4 and H2O HETATM records. Since each segment must have a separate input file, we remove all non-protein atom records using grep. If there were multiple chains we would have to split the file by hand.

```
grep -v '^HETATM' 6PTI.pdb > output/6PTI_protein.pdb
```

Create a second file containing only waters.

```
grep 'HOH' 6PTI.pdb > output/6PTI_water.pdb
```

Run the psfgen program, taking everything until "ENDMOL" as input. You may run psfgen interactively as well. Since psfgen is built on a Tcl interpreter, you may use loops, variables, etc., but you must use $$ for variables when inside a shell script. If you want, run psfgen and enter the following commands manually.

```
psfgen << ENDMOL
```

Reading topology file. Read in the topology definitions for the residues we will create. This must match the parameter file used for the simulation as well. Multiple topology files may be read in since psfgen and NAMD use atom type names rather than numbers in psf files.

```
topology toppar/top_all22_prot.inp
```

Building segment BPTI. Actually build a segment, calling it BPTI and reading the sequence of residues from the stripped pdb file created above. In addition to the pdb command, we could specify residues explicitly. Both angles and dihedrals are generated automatically unless "auto none" is added (which is required to build residues of water). The commands "first" and "last" may be used to change the default patches for the ends of the chain. The structure is built when the closing } is encountered, and some errors regarding the first and last residue are normal.

```
segment BPTI {
 pdb output/6PTI_protein.pdb
}
```

Adding patches. Some patch residues (those not used to begin or end a chain) are applied after the segment is built. These contain all angle and dihedral terms explicitly since they were already generated. In this case we apply the patch for a disulfide link three separate times.

```
patch DISU BPTI:5 BPTI:55
patch DISU BPTI:14 BPTI:38
patch DISU BPTI:30 BPTI:51
```

Reading coordinates from pdb file. The same file used to generate the sequence is now read to extract coordinates. In the residue ILE, the atom CD is called CD1 in the pdb file, so we use "alias atom" to define the correct name. Segment names in the pdb file are ignored so we specify that the coordinates should be applied to the segment BPTI.

```
alias atom ILE CD1 CD
coordpdb output/6PTI_protein.pdb BPTI
```

Adding a segment of water. Build a segment for the crystal waters. The residue type for water depends on the model, so here we alias HOH to TIP3. Because CHARMM uses an additional H-H bond we must disable generation of angles and dihedrals for segments containing water. Then read the pdb file.

```
alias residue HOH TIP3
segment SOLV {
 auto none
 pdb output/6PTI_water.pdb
}
```

Reading water coordinates. Alias the atom type for water oxygen as well and read coordinates from the file to the segment SOLV. Hydrogen doesn't show up in crystal structures so it is missing from this pdb file.

```
alias atom HOH O OH2
coordpdb output/6PTI_water.pdb SOLV
```

Writing psf structure file. Now that all of the atoms and bonds have been created, we can write out the psf structure file for the system.

```
writepsf output/bpti.psf
```

Guessing missing coordinates. The tolopogy file contains default internal coordinates which can be used to guess the locations of many atoms, hydrogens in particular. In the output pdb file, the occupancy field of guessed atoms will be set to 0, atoms which are known are set to 1, and atoms which could not be guessed are set to -1. Some atoms are "poorly guessed" if needed bond lengths and angles were missing from the topology file. Similarly, waters with missing hydrogen coordinates are given a default orientation.

```
guesscoord
```

Writing pdb coordinate file. This creates the matching coordinate pdb file. The psf and pdb files are a matched set with identical atom ordering as needed by NAMD.

```
writepdb output/bpti.pdb

ENDMOL
```

Using generated files in NAMD. The files bpti.pdb and bpti.psf can now be used with NAMD, but the initial coordinates require minimization first. The following is an example NAMD configuration file for the BPTI example.

```
# NAMD configuration file for BPTI

# molecular system
structure output/bpti.psf

# force field
paratypecharmm on
parameters toppar/par_all22_prot.inp
exclude scaled1-4
1-4scaling 1.0

# approximations
switching on
switchdist 8
cutoff 12
pairlistdist 13.5
margin 0
stepspercycle 20

#integrator
timestep 1.0

#output
outputenergies 10
outputtiming 100
binaryoutput no
```

```
# molecular system
coordinates output/bpti.pdb

#output
outputname output/bpti
dcdfreq 1000

#protocol
temperature 0
reassignFreq 1000
reassignTemp 25
reassignIncr 25
reassignHold 300

#script

minimize 1000

run 20000
```

4.4 Building solvent around a protein

The following script illustrates how psfgen and VMD can be used together to add water around a protein structure. It assumes you already have a psf and pdb file for your protein, as well as a box of water which is large enough to contain the protein. For more information on how atomselections can be used within VMD scripts, see the VMD User's Guide.

```
proc addwater { psffile pdbfile watpsf watpdb } {
# Create psf/pdb files that contain both our protein as well as
# a box of equilibrated water.  The water box should be large enough
# to easily contain our protein.
resetpsf
readpsf $psffile
readpsf $watpsf
coordpdb $pdbfile
coordpdb $watpdb

# Load the combined structure into VMD
writepsf combine.psf
writepdb combine.pdb
mol load psf combine.psf pdb combine.pdb

# Assume that the segid of the water in watpsf is QQQ
# We want to delete waters outside of a box ten Angstroms
# bigger than the extent of the protein.
set protein [atomselect top "not segid QQQ"]
set minmax [measure minmax $protein]
foreach {min max} $minmax { break }
foreach {xmin ymin zmin} $min { break }
foreach {xmax ymax zmax} $max { break }
    set xmin [expr $xmin - 10]
    set ymin [expr $ymin - 10]
    set zmin [expr $zmin - 10]
    set xmax [expr $xmax + 10]
    set ymax [expr $ymax + 10]
    set zmax [expr $zmax + 10]

# Center the water on the protein.  Also update the coordinates held
# by psfgen.
set wat [atomselect top "segid QQQ"]
$wat moveby [vecsub [measure center $protein] [measure center $wat]]
foreach atom [$wat get {segid resid name x y z}] {
foreach {segid resid name x y z} $atom { break }
coord $segid $resid $name [list $x $y $z]
}

# Select waters that we don't want in the final structure.
set outsidebox [atomselect top "segid QQQ and (x <= $xmin or y <= $ymin \
or z <= $zmin or x >= $xmax or y >= $ymax or z >= $xmax)"]
set overlap [atomselect top "segid QQQ and within 2.4 of (not segid QQQ)"]

# Get a list of all the residues that are in the two selections, and delete
# those residues from the structure.
set reslist [concat [$outsidebox get resid] [$overlap get resid]]
set reslist [lsort -unique -integer $reslist]

foreach resid $reslist {
delatom QQQ $resid
}

# That should do it - write out the new psf and pdb file.
writepsf solvate.psf
writepdb solvate.pdb

# Delete the combined water/protein molecule and load the system that
# has excess water removed.
mol delete top
mol load psf solvate.psf pdb solvate.pdb

# Return the size of the water box
return [list [list $xmin $ymin $zmin] [list $xmax $ymax $zmax]]
}
```

5 Basic Simulation Parameters

5.1 Non-bonded interaction parameters and computations

NAMD has a number of options that control the way that non-bonded interactions are calculated. These options are interrelated and can be quite confusing, so this section attempts to explain the behavior of the non-bonded interactions and how to use these parameters.

5.1.1 Non-bonded van der Waals interactions

The simplest non-bonded interaction is the van der Waals interaction. In NAMD, van der Waals interactions are always truncated at the cutoff distance, specified by `cutoff`. The main option that effects van der Waals interactions is the `switching` parameter. With this option set to `on`, a smooth switching function will be used to truncate the van der Waals potential energy smoothly at the cutoff distance. A graph of the van der Waals potential with this switching function is shown in Figure 1. If `switching` is set to `off`, the van der Waals energy is just abruptly truncated at the cutoff distance, so that energy may not be conserved.

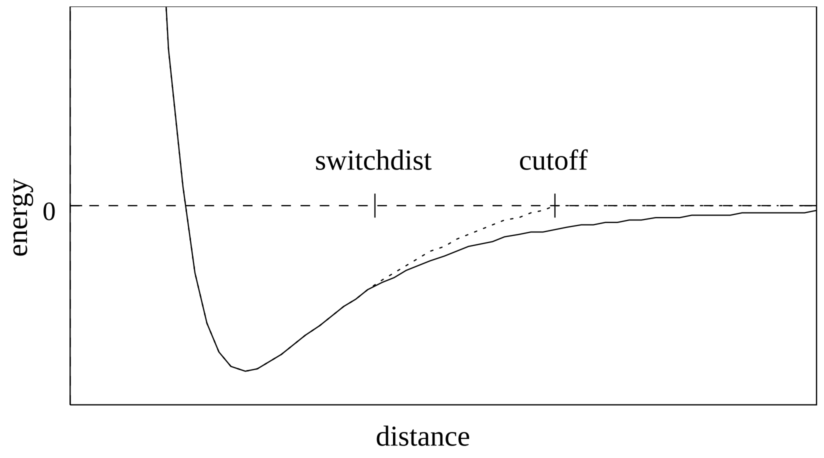

Figure 1: Graph of van der Waals potential with and without the application of the switching function. With the switching function active, the potential is smoothly reduced to 0 at the cutoff distance. Without the switching function, there is a discontinuity where the potential is truncated.

The switching function used is based on the X-PLOR switching function. The parameter `switchdist` specifies the distance at which the switching function should start taking effect to bring the van der Waals potential to 0 smoothly at the cutoff distance. Thus, the value of `switchdist` must always be less than that of `cutoff`.

5.1.2 Non-bonded electrostatic interactions

The handling of electrostatics is slightly more complicated due to the incorporation of multiple timestepping for full electrostatic interactions. There are two cases to consider, one where full electrostatics is employed and the other where electrostatics are truncated at a given distance.

First let us consider the latter case, where electrostatics are truncated at the cutoff distance. Using this scheme, all electrostatic interactions beyond a specified distance are ignored, or assumed to be zero. If `switching` is set to `on`, rather than having a discontinuity in the potential at the cutoff distance, a shifting function is applied to the electrostatic potential as shown in Figure 2. As this figure shows, the shifting function shifts the entire potential curve so that the curve intersects the x-axis at the cutoff distance. This shifting function is based on the shifting function used by X-PLOR.

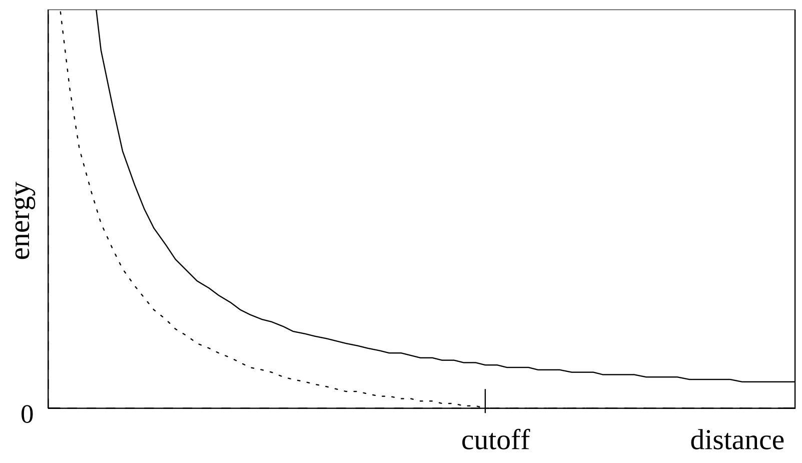

Figure 2: Graph showing an electrostatic potential with and without the application of the shifting function.

Next, consider the case where full electrostatics are calculated. In this case, the electrostatic interactions are not truncated at any distance. In this scheme, the `cutoff` parameter has a slightly different meaning for the electrostatic interactions — it represents the *local interaction distance*, or distance within which electrostatic pairs will be directly calculated every timestep. Outside of this distance, interactions will be calculated only periodically. These forces will be applied using a multiple timestep integration scheme as described in Section 5.2.

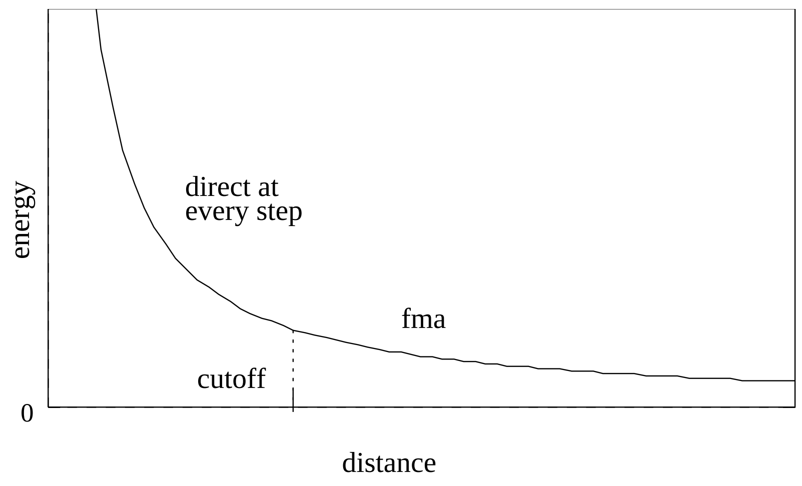

Figure 3: Graph showing an electrostatic potential when full electrostatics are used within NAMD, with one curve portion calculated directly and the other calculated using DPMTA.

5.1.3 Nonbonded interaction distance-testing

The last critical parameter for non-bonded interaction calculations is the parameter `pairlistdist`. To reduce the cost of performing the non-bonded interactions, NAMD 1.X used a *non-bonded pair list* which contained all pairs of atoms for which non-bonded interactions should be calculated. Performing the search for pairs of atoms that should have their interactions calculated is an expensive operation. Thus, the pair list is only calculated periodically, once per cycle. Unfortunately, pairs of atoms move relative to each other during the steps between preparation of the pair list. Because of this, if the pair list were built to include only those pairs of atoms that are within the cutoff distance when the list is generated, it would be possible for atoms to drift closer together than the cutoff distance during subsequent timesteps and yet not have their non-bonded interactions calculated.

Let us consider a concrete example to better understand this. Assume that the pairlist is built once every ten timesteps and that the cutoff distance is 8.0 Å. Consider a pair of atoms A and B that are 8.1 Å apart when the pairlist is built. If the pair list includes only those atoms within the cutoff distance, this pair would not be included in the list. Now assume that after five timesteps, atoms A and B have moved to only 7.9 Å apart. A and B are now within the cutoff distance of each other, and should have their non-bonded interactions calculated. However, because the non-bonded interactions are based solely on the pair list and the pair list will not be rebuilt for another five timesteps, this pair will be ignored for five timesteps causing energy not to be conserved within the system.

To avoid this problem, the parameter `pairlistdist` allowed the user to specify a distance greater than the `cutoff` distance for pairs to be included in the pair list, as shown in Figure 4. Pairs that are included in the pair list but are outside the cutoff distance are simply ignored. So in the above example, if the `pairlistdist` were set to 10.0 Å, then the atom pair A and B would be included in the pair list, even though the pair would initially be ignored because they are further apart than the cutoff distance. As the pair moved closer and entered the cutoff distance, because the pair was already in the pair list, the non-bonded interactions would immediately be calculated and energy conservation would be preserved. The value of `pairlistdist` should be chosen such that no atom pair moves more than `pairlistdist` − `cutoff` in one cycle. This will insure energy conservation.

NAMD 2.X eliminated the explicit use of pairlists in order to reduce memory usage in light of equally efficient distance-testing algorithms. Specifically, it was realized that building a pairlist on top of the existing spatial decomposition was only marginally more efficient than actually testing atom distances at every timestep given efficient methods for dealing with nonbonded exclusions. The `pairlistdist` parameter now serves the same function, but is instead used to determine the minimum patch size. Unless the `splitPatch` parameter is explicitly set to `position`, hydrogen atoms will be placed on the same patch as the "mother atom" to which they are bonded. These *hydrogen groups* are then distance tested against each other using only a cutoff increased by the the value of the `hgroupCutoff` parameter. The size of the patches is also increased by this amount. NAMD functions correctly even if a hydrogen atom and its mother atom are separated by more than half of `hgroupCutoff` by breaking that group into its individual atoms for distance testing. Warning messages are printed if an atom moves outside of a safe zone surrounding the patch to which it is assigned, indicating that `pairlistdist` should be increased in order for forces to be calculated correctly and energy to be conserved.

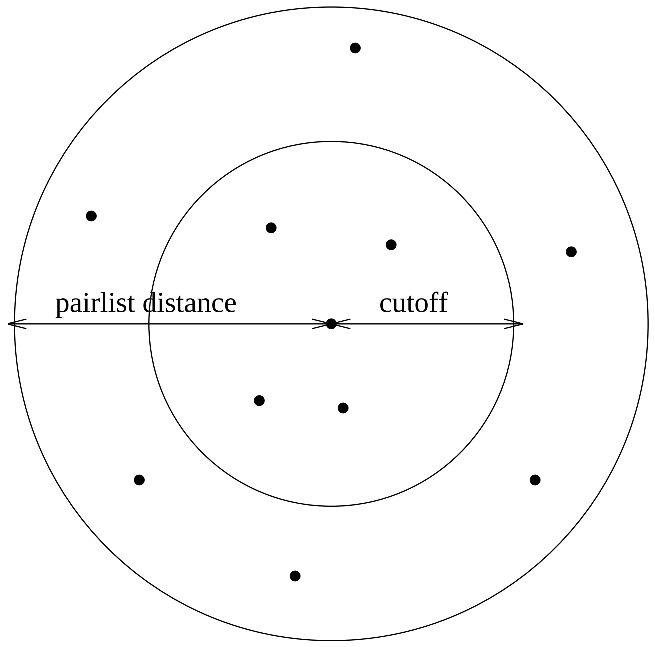

Figure 4: Depiction of the difference between the cutoff distance and the pair list distance. The pair list distance specifies a sphere that is slightly larger than that of the cutoff so that pairs are allowed to move in and out of the cutoff distance without causing energy conservation to be disturbed.

5.2 Full electrostatic integration

To further reduce the cost of computing full electrostatics, NAMD uses a multiple timestepping integration scheme. In this scheme, the total force acting on each atom is broken into two pieces, a quickly varying local component and a slower long range component. The local force component is defined in terms of a *splitting function.* The local force component consists of all bonded and van der Waals interactions as well as that portion of electrostatic interactions for pairs that are separated by less than the local interaction distance determined by the splitting function. The long range component consists only of electrostatic interactions outside of the local interaction distance. Since the long range forces are slowly varying, they are not evaluated every timestep. Instead, they are evaluated every k timesteps, specified by the NAMD parameter `fullElectFrequency`. An impulse of k times the long range force is applied to the system every k timesteps (i.e., the r-RESPA integrator is used). For appropriate values of k, it is believed that the error introduced by this infrequent evaluation is modest compared to the error already incurred by the use of the numerical (Verlet) integrator. Improved methods for incorporating these long range forces are currently being investigated, with the intention of improving accuracy as well as reducing the frequency of long range force evaluations.

In the scheme described above, the van der Waals forces are still truncated at the local interaction distance. Thus, the van der Waals cutoff distance forms a lower limit to the local interaction distance. While this is believed to be sufficient, there are investigations underway to remove this limitation and provide full van der Waals calculations in $\mathcal{O}(N)$ time as well.

5.3 NAMD configuration parameters

5.3.1 Timestep parameters

- `numsteps` < number of timesteps >
  Acceptable Values: positive integer
  Description: The number of simulation timesteps to be performed. An integer greater than 0 is acceptable. The total amount of simulation time is `numsteps` $\times$ `timestep`.

- `timestep` < timestep size (fs) >
  Acceptable Values: non-negative decimal
  Default Value: 1.0
  Description: The timestep size to use when integrating each step of the simulation. The value is specified in femtoseconds.

- `firsttimestep` < starting timestep value >
  Acceptable Values: non-negative integer
  Default Value: 0
  Description: The number of the first timestep. This value is typically used only when a simulation is a continuation of a previous simulation. In this case, rather than having the timestep restart at 0, a specific timestep number can be specified.

- `stepspercycle` < timesteps per cycle >
  Acceptable Values: positive integer
  Default Value: 20
  Description: Number of timesteps in each cycle. Each cycle represents the number of timesteps between atom reassignments. For more details on non-bonded force evaluation, see Section 5.1.

5.3.2 Simulation space partitioning

- `cutoff` < local interaction distance common to both electrostatic and van der Waals calculations (Å) >
  Acceptable Values: positive decimal
  Description: See Section 5.1 for more information.

- `switching` < use switching function? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: If `switching` is specified to be `off`, then a truncated cutoff is performed. If `switching` is turned `on`, then smoothing functions are applied to both the electrostatics and van der Waals forces. For a complete description of the non-bonded force parameters see Section 5.1. If `switching` is set to `on`, then `switchdist` must also be defined.

- `switchdist` < distance at which to activate switching function for electrostatic and van der Waals calculations (Å) >
  Acceptable Values: positive decimal $\leq$ `cutoff`
  Description: Distance at which the switching function should begin to take effect. This parameter only has meaning if `switching` is set to `on`. The value of `switchdist` must be less than or equal to the value of `cutoff`, since the switching function is only applied on the range

from `switchdist` to `cutoff`. For a complete description of the non-bonded force parameters see Section 5.1.

- `pairlistdist` < distance between pairs for inclusion in pair lists (Å) >
  Acceptable Values: positive decimal ≥ `cutoff`
  Default Value: `cutoff`
  Description: A pair list is generated each cycle, containing pairs of atoms for which electrostatics and van der Waals interactions will be calculated. This parameter is used when `switching` is set to `on` to specify the allowable distance between atoms for inclusion in the pair list. This parameter is equivalent to the X-PLOR parameter `CUTNb`. If no atom moves more than `pairlistdist`−`cutoff` during one cycle, then there will be no jump in electrostatic or van der Waals energies when the next pair list is built. Since such a jump is unavoidable when truncation is used, this parameter may only be specified when `switching` is set to `on`. If this parameter is not specified and `switching` is set to `on`, the value of `cutoff` is used. A value of at least one greater than `cutoff` is recommended.

- `splitPatch` < how to assign atoms to patches >
  Acceptable Values: `position` or `hydrogen`
  Default Value: `hydrogen`
  Description: When set to `hydrogen`, hydrogen atoms are kept on the same patch as their parents, allowing faster distance checking and rigid bonds.

- `hgroupCutoff (Å)` < used for group-based distance testing >
  Acceptable Values: positive decimal
  Default Value: 2.5
  Description: This should be set to twice the largest distance which will ever occur between a hydrogen atom and its mother. Warnings will be printed if this is not the case. This value is also added to the margin.

- `margin` < extra length in patch dimension (Å) >
  Acceptable Values: positive decimal
  Default Value: 0.0
  Description: An internal tuning parameter used in determining the size of the cubes of space with which NAMD uses to partition the system. The value of this parameter will not change the physical results of the simulation. Unless you are very motivated to get the *very* best possible performance, just leave this value at the default.

5.3.3 Basic dynamics

- `exclude` < exclusion policy to use >
  Acceptable Values: `none`, `1-2`, `1-3`, `1-4`, or `scaled1-4`
  Description: This parameter specifies which pairs of bonded atoms should be excluded from non-bonded interactions. With the value of `none`, no bonded pairs of atoms will be excluded. With the value of `1-2`, all atom pairs that are directly connected via a linear bond will be excluded. With the value of `1-3`, all `1-2` pairs will be excluded along with all pairs of atoms that are bonded to a common third atom (i.e., if atom A is bonded to atom B and atom B is bonded to atom C, then the atom pair A-C would be excluded). With the value of `1-4`, all `1-3` pairs will be excluded along with all pairs connected by a set of two bonds (i.e.,

if atom A is bonded to atom B, and atom B is bonded to atom C, and atom C is bonded to atom D, then the atom pair A-D would be excluded). With the value of `scaled1-4`, all `1-3` pairs are excluded and all pairs that match the `1-4` criteria are modified. The electrostatic interactions for such pairs are modified by the constant factor defined by `1-4scaling`. The van der Waals interactions are modified by using the special 1-4 parameters defined in the parameter files.

- `temperature` < initial temperature (K) >
  Acceptable Values: positive decimal
  Description: Initial temperature value for the system. Using this option will generate a random velocity distribution for the initial velocities for all the atoms such that the system is at the desired temperature. Either the `temperature` or the `velocities/binvelocities` option must be defined to determine an initial set of velocities. Both options cannot be used together.

- `COMmotion` < allow center of mass motion? >
  Acceptable Values: `yes` or `no`
  Default Value: `no`
  Description: Specifies whether or not motion of the center of mass of the entire system is allowed. If this option is set to `no`, the initial velocities of the system will be adjusted to remove center of mass motion of the system. Note that this does not preclude later center-of-mass motion due to external forces such as random noise in Langevin dynamics, boundary potentials, and harmonic restraints.

- `dielectric` < dielectric constant for system >
  Acceptable Values: decimal ≥ 1.0
  Default Value: 1.0
  Description: Dielectric constant for the system. A value of 1.0 implies no modification of the electrostatic interactions. Any larger value will lessen the electrostatic forces acting in the system.

- `1-4scaling` < scaling factor for 1-4 interactions >
  Acceptable Values: $0 \leq$ decimal ≤ 1
  Default Value: 1.0
  Description: Scaling factor for 1-4 interactions. This factor is only used when the `exclude` parameter is set to `scaled1-4`. In this case, this factor is used to modify the electrostatic interactions between 1-4 atom pairs. If the `exclude` parameter is set to anything but `scaled1-4`, this parameter has no effect regardless of its value.

- `seed` < random number seed >
  Acceptable Values: positive integer
  Default Value: pseudo-random value based on current UNIX clock time
  Description: Number used to seed the random number generator if `temperature` or `langevin` is selected. This can be used so that consecutive simulations produce the same results. If no value is specified, NAMD will choose a pseudo-random value based on the current UNIX clock time. The random number seed will be output during the simulation startup so that its value is known and can be reused for subsequent simulations. Note that if Langevin dynamics are used in a parallel simulation (i.e., a simulation using more than one processor) even using the same seed will *not* guarantee reproducible results.

- `rigidBonds` < controls if and how ShakeH is used >
  Acceptable Values: `none`, `water`, `all`
  Default Value: `none`
  Description: When `rigidBonds` is `all`, the bond between each hydrogen and its mother atom is fixed to the nominal bond length given in the parameter file. When `water` is selected, only the bonds between the hydrogens and the oxygen in water molecules are constrained. For the default case `none`, no lengths are constrained.

- `rigidTolerance` < allowable bond-length error for ShakeH (Å) >
  Acceptable Values: positive decimal
  Default Value: 0.00001
  Description: The ShakeH algorithm is assumed to have converged when all constrained bonds differ from the nominal bond length by less than this amount.

- `rigidIterations` < maximum ShakeH iterations >
  Acceptable Values: positive integer
  Default Value: 100
  Description: The maximum number of iterations ShakeH will perform before giving up on constraining the bond lengths. If the bond lengths do not converge, a warning message is printed, and the atoms are left at the final value achieved by ShakeH. Although the default value is 100, convergence is usually reached after fewer than 10 iterations.

5.3.4 DPMTA parameters

DPMTA is no longer included in the released NAMD binaries. We recommend that you instead use PME with a periodic system because it conserves energy better, is more efficient, and is better parallelized. If you must have the fast multipole algorithm you may compile NAMD yourself.

These parameters control the options to DPMTA, an algorithm used to provide full electrostatic interactions. DPMTA is a modified version of the FMA (Fast Multipole Algorithm) and, unfortunately, most of the parameters still refer to FMA rather than DPMTA for historical reasons. Don't be confused!

For a further description of how exactly full electrostatics are incorporated into NAMD, see Section 5.2. For a greater level of detail about DPMTA and the specific meaning of its options, see the DPMTA distribution which is available via anonymous FTP from the site `ftp.ee.duke.edu` in the directory `/pub/SciComp/src`.

- `FMA` < use full electrostatics? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not the DPMTA algorithm from Duke University should be used to compute the full electrostatic interactions. If set to `on`, DPMTA will be used with a multiple timestep integration scheme to provide full electrostatic interactions as detailed in Section 5.2. *DPMTA is no longer included in released binaries.*

- `FMALevels` < number of levels to use in multipole expansion >
  Acceptable Values: positive integer
  Default Value: 5
  Description: Number of levels to use for the multipole expansion. This parameter is only

used if `FMA` is set to `on`. A value of 4 should be sufficient for systems with less than 10,000 atoms. A value of 5 or greater should be used for larger systems.

- `FMAMp` < number of multipole terms to use for FMA >
  Acceptable Values: positive integer
  Default Value: 8
  Description: Number of terms to use in the multipole expansion. This parameter is only used if `FMA` is set to `on`. If the `FMAFFT` is set to `on`, then this value must be a multiple of 4. The default value of 8 should be suitable for most applications.

- `FMAFFT` < use DPMTA FFT enhancement? >
  Acceptable Values: `on` or `off`
  Default Value: `on`
  Description: Specifies whether or not the DPMTA code should use the FFT enhancement feature. This parameter is only used if `FMA` is set to `on`. If `FMAFFT` is set to `on`, the value of `FMAMp` must be set to a multiple of 4. This feature offers substantial benefits only for values of `FMAMp` of 8 or greater. This feature will substantially increase the amount of memory used by DPMTA.

- `FMAtheta` < DPMTA theta parameter (radians) >
  Acceptable Values: decimal
  Default Value: 0.715
  Description: This parameter specifies the value of the theta parameter used in the DPMTA calculation. The default value is based on recommendations by the developers of the code.

- `FMAFFTBlock` < blocking factor for FMA FFT >
  Acceptable Values: positive integer
  Default Value: 4
  Description: The blocking factor for the FFT enhancement to DPMTA. This parameter is only used if both `FMA` and `FMAFFT` are set to `on`. The default value of 4 should be suitable for most applications.

5.3.5 PME parameters

PME stands for Particle Mesh Ewald and is an efficient full electrostatics method for use with periodic boundary conditions. None of the parameters should affect energy conservation, although they may affect the accuracy of the results and momentum conservation.

- `PME` < Use particle mesh Ewald for electrostatics? >
  Acceptable Values: `yes` or `no`
  Default Value: `no`
  Description: Turns on particle mesh Ewald.

- `PMETolerance` < PME direct space tolerance >
  Acceptable Values: positive decimal
  Default Value: 10^{-6}
  Description: Affects the value of the Ewald coefficient and the overall accuracy of the results.

- `PMEInterpOrder` < PME interpolation order >
  Acceptable Values: positive integer
  Default Value: 4 (cubic)
  Description: Charges are interpolated onto the grid and forces are interpolated off using this many points, equal to the order of the interpolation function plus one.

- `PMEGridSizeX` < number of grid points in x dimension >
  Acceptable Values: positive integer
  Description: The grid size partially determines the accuracy and efficiency of PME. For speed, `PMEGridSizeX` should have only small integer factors (2, 3 and 5).

- `PMEGridSizeY` < number of grid points in y dimension >
  Acceptable Values: positive integer
  Description: The grid size partially determines the accuracy and efficiency of PME. For speed, `PMEGridSizeY` should have only small integer factors (2, 3 and 5).

- `PMEGridSizeZ` < number of grid points in z dimension >
  Acceptable Values: positive integer
  Description: The grid size partially determines the accuracy and efficiency of PME. For speed, `PMEGridSizeZ` should have only small integer factors (2, 3 and 5).

- `PMEProcessors` < processors for FFT and reciprocal sum >
  Acceptable Values: positive integer
  Default Value: larger of x and y grid sizes up to all available processors
  Description: For best performance on some systems and machines, it may be necessary to restrict the amount of parallelism used. Experiment with this parameter if your parallel performance is poor when PME is used.

- `useDPME` < Use old DPME code? >
  Acceptable Values: `yes` or `no`
  Default Value: `no`
  Description: Switches to old DPME implementation of particle mesh Ewald. The new code is faster and allows non-orthogonal cells so you probably just want to leave this option turned off. If you set `cellOrigin` to something other than $(0, 0, 0)$ the energy may differ slightly between the old and new implementations. *DPME is no longer included in released binaries.*

5.3.6 Full direct parameters

The direct computation of electrostatics is not intended to be used during real calculations, but rather as a testing or comparison measure. Because of the $\mathcal{O}(N^2)$ computational complexity for performing direct calculations, this is *much* slower than using DPMTA or PME to compute full electrostatics for large systems. In the case of periodic boundary conditions, the nearest image convention is used rather than a full Ewald sum.

- `FullDirect` < calculate full electrostatics directly? >
  Acceptable Values: `yes` or `no`
  Default Value: `no`
  Description: Specifies whether or not direct computation of full electrostatics should be performed.

5.3.7 Multiple timestep parameters

One of the areas of current research being studied using NAMD is the exploration of better methods for performing multiple timestep integration. Currently the only available method is the impulse-based Verlet-I or r-RESPA method which is stable for timesteps up to 4 fs for long-range electrostatic forces, 2 fs for short-range nonbonded forces, and 1 fs for bonded forces Setting `rigid all` (i.e., using SHAKE) increases these timesteps to 6 fs, 2 fs, and 2 fs respectively but eliminates bond motion for hydrogen. The mollified impulse method (MOLLY) reduces the resonance which limits the timesteps and thus increases these timesteps to 6 fs, 2 fs, and 1 fs while retaining all bond motion.

- `fullElectFrequency` < number of timesteps between full electrostatic evaluations >
  Acceptable Values: positive integer factor of `stepspercycle`
  Default Value: `nonbondedFreq`
  Description: This parameter specifies the number of timesteps between each full electrostatics evaluation. It is recommended that `fullElectFrequency` be chosen so that the product of `fullElectFrequency` and `timestep` does not exceed 4.0 unless `rigidBonds all` or `molly on` is specified, in which case the upper limit is perhaps doubled.

- `nonbondedFreq` < timesteps between nonbonded evaluation >
  Acceptable Values: positive integer factor of `fullElectFrequency`
  Default Value: 1
  Description: This parameter specifies how often short-range nonbonded interactions should be calculated. Setting `nonbondedFreq` between 1 and `fullElectFrequency` allows triple timestepping where, for example, one could evaluate bonded forces every 1 fs, short-range nonbonded forces every 2 fs, and long-range electrostatics every 4 fs.

- `MTSAlgorithm` < MTS algorithm to be used >
  Acceptable Values: `impulse/verletI` or `constant/naive`
  Default Value: `impulse`
  Description: Specifies the multiple timestep algorithm used to integrate the long and short range forces. `impulse/verletI` is the same as r-RESPA. `constant/naive` is the stale force extrapolation method.

- `longSplitting` < how should long and short range forces be split? >
  Acceptable Values: `xplor`, `c1`
  Default Value: `c1`
  Description: Specifies the method used to split electrostatic forces between long and short range potentials. The `xplor` option uses the X-PLOR shifting function, and the `c1` splitting uses the following C^1 continuous shifting function [10]:

  $SW(r_{ij}) = 0$ if $|\vec{r}_{ij}| > R_{off}$

  $SW(r_{ij}) = 1$ if $|\vec{r}_{ij}| \leq R_{on}$

  if $R_{off} > |\vec{r}_{ij}| \geq R_{on}$

  where

  R_{on} is a constant defined using the configuration value `switchdist`

  R_{off} is specified using the configuration value `cutoff`

- `molly` < use mollified impulse method (MOLLY)? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: This method eliminates the components of the long range electrostatic forces which contribute to resonance along bonds to hydrogen atoms, allowing a fullElectFrequency of 6 (vs. 4) with a 1 fs timestep without using `rigidBonds all`. You may use `rigidBonds water` but using `rigidBonds all` with MOLLY makes no sense since the degrees of freedom which MOLLY protects from resonance are already frozen.

- `mollyTolerance` < allowable error for MOLLY >
  Acceptable Values: positive decimal
  Default Value: 0.00001
  Description: Convergence criterion for MOLLY algorithm.

- `mollyIterations` < maximum MOLLY iterations >
  Acceptable Values: positive integer
  Default Value: 100
  Description: Maximum number of iterations for MOLLY algorithm.

6 Additional Simulation Parameters

6.1 Constraints and Restraints

6.1.1 Harmonic constraint parameters

The following describes the parameters for the harmonic constraints feature of NAMD. Actually, this feature should be referred to as harmonic restraints rather than constraints, but for historical reasons the terminology of harmonic constraints has been carried over from X-PLOR. This feature allows a harmonic restraining force to be applied to any set of atoms in the simulation.

- `constraints` < are constraints active? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not harmonic constraints are active. If it is set to `off`, then no harmonic constraints are computed. If it is set to `on`, then harmonic constraints are calculated using the values specified by the parameters `consref`, `conskfile`, `conskcol`, and `consexp`.

- `consexp` < exponent for harmonic constraint energy function >
  Acceptable Values: positive, even integer
  Default Value: 2
  Description: Exponent to be use in the harmonic constraint energy function. This value must be a positive integer, and only even values really make sense. This parameter is used only if `constraints` is set to `on`.

- `consref` < PDB file containing constraint reference positions >
  Acceptable Values: UNIX file name
  Default Value: `coordinates`
  Description: PDB file to use for reference positions for harmonic constraints. Each atom that has an active constraint will be constrained about the position specified in this file. If no value is given and constraints are active, then the same PDB file specified by `coordinates` will be used instead, constraining atoms about their initial positions.

- `conskfile` < PDB file containing force constant values >
  Acceptable Values: UNIX filename
  Default Value: `coordinates`
  Description: PDB file to use for force constants for harmonic constraints. If this parameter is not specified, then the PDB file containing initial coordinates specified by `coordinates` is used.

- `conskcol` < column of PDB file containing force constant >
  Acceptable Values: `X`, `Y`, `Z`, `O`, or `B`
  Default Value: `O`
  Description: Column of the PDB file to use for the harmonic constraint force constant. This parameter may specify any of the floating point fields of the PDB file, either X, Y, Z, occupancy, or beta-coupling (temperature-coupling). Regardless of which column is used, a value of 0 indicates that the atom should not be constrained. Otherwise, the value specified is used as the force constant for that atom's restraining potential.

- `selectConstraints` < Restrain only selected Cartesian components of the coordinates? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: This option is useful to restrain the positions of atoms to a plane or a line in space. If active, this option will ensure that only selected Cartesian components of the coordinates are restrained. E.g.: Restraining the positions of atoms to their current z values with no restraints in x and y will allow the atoms to move in the x-y plane while retaining their original z-coordinate. Restraining the x and y values will lead to free motion only along the z coordinate.

- `selectConstrX` < Restrain X components of coordinates >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Restrain the Cartesian x components of the positions.

- `selectConstrY` < Restrain Y components of coordinates >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Restrain the Cartesian y components of the positions.

- `selectConstrZ` < Restrain Z components of coordinates >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Restrain the Cartesian z components of the positions.

6.1.2 Fixed atoms parameters

Atoms may be held fixed during a simulation. NAMD avoids calculating most interactions in which all affected atoms are fixed unless `fixedAtomsForces` is specified.

- `fixedAtoms` < are there fixed atoms? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not fixed atoms are present.

- `fixedAtomsForces` < are forces between fixed atoms calculated? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not forces between fixed atoms are calculated. This option is required for constant pressure or to turn fixed atoms off in the middle of a simulation.

- `fixedAtomsFile` < PDB file containing fixed atom parameters >
  Acceptable Values: UNIX filename
  Default Value: `coordinates`
  Description: PDB file to use for the fixed atom flags for each atom. If this parameter is not specified, then the PDB file specified by `coordinates` is used.

- `fixedAtomsCol` < column of PDB containing fixed atom parameters >
  Acceptable Values: `X`, `Y`, `Z`, `O`, or `B`

Default Value: `0`
Description: Column of the PDB file to use for the containing fixed atom parameters for each atom. The coefficients can be read from any floating point column of the PDB file. A value of 0 indicates that the atom is not fixed.

6.2 Energy Minimization

6.2.1 Conjugate gradient parameters

The default minimizer uses a sophisticated conjugate gradient and line search algorithm with much better performance than the older velocity quenching method. The method of conjugate gradients is used to select successive search directions (starting with the initial gradient) which eliminate repeated minimization along the same directions. Along each direction, a minimum is first bracketed (rigorously bounded) and then converged upon by either a golden section search, or, when possible, a quadratically convergent method using gradient information.

For most systems, it just works.

- `minimization` < Perform conjugate gradient energy minimization? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Turns efficient energy minimization `on` or `off`.

- `minTinyStep` < first initial step for line minimizer >
  Acceptable Values: positive decimal
  Default Value: 1.0e-6
  Description: If your minimization is immediately unstable, make this smaller.

- `minBabyStep` < max initial step for line minimizer >
  Acceptable Values: positive decimal
  Default Value: 1.0e-2
  Description: If your minimization becomes unstable later, make this smaller.

- `minLineGoal` < gradient reduction factor for line minimizer >
  Acceptable Values: positive decimal
  Default Value: 1.0e-4
  Description: Varying this might improve conjugate gradient performance.

6.2.2 Velocity quenching parameters

You can perform energy minimization using a simple quenching scheme. While this algorithm is not the most rapidly convergent, it is sufficient for most applications. There are only two parameters for minimization: one to activate minimization and another to specify the maximum movement of any atom.

- `velocityQuenching` < Perform old-style energy minimization? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Turns slow energy minimization `on` or `off`.

- `maximumMove` < maximum distance an atom can move during each step (Å) >
  Acceptable Values: positive decimal
  Default Value: $0.75 \times$ `cutoff`/`stepsPerCycle`
  Description: Maximum distance that an atom can move during any single timestep of minimization. This is to insure that atoms do not go flying off into space during the first few timesteps when the largest energy conflicts are resolved.

6.3 Temperature Control and Equilibration

6.3.1 Langevin dynamics parameters

NAMD is capable of performing Langevin dynamics, where additional damping and random forces are introduced to the system. This capability is based on that implemented in X-PLOR which is detailed in the X-PLOR *User's Manual* [7], although a different integrator is used.

- `langevin` < use Langevin dynamics? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not Langevin dynamics active. If set to `on`, then the parameter `langevinTemp` must be set and the parameters `langevinFile` and `langevinCol` can optionally be set to control the behavior of this feature.

- `langevinTemp` < temperature for Langevin calculations (K) >
  Acceptable Values: positive decimal
  Description: Temperature to which atoms affected by Langevin dynamics will be adjusted. This temperature will be roughly maintained across the affected atoms through the addition of friction and random forces.

- `langevinDamping` < damping coefficient for Langevin dynamics (1/ps) >
  Acceptable Values: positive decimal
  Default Value: per-atom values from PDB file
  Description: Langevin coupling coefficient to be applied to all atoms (unless `langevinHydrogen` is `off`, in which case only non-hydrogen atoms are affected). If not given, a PDB file is used to obtain coefficients for each atom (see `langevinFile` and `langevinCol` below).

- `langevinHydrogen` < Apply Langevin dynamics to hydrogen atoms? >
  Acceptable Values: `on` or `off`
  Default Value: `on`
  Description: If `langevinDamping` is set then setting `langevinHydrogen` to `off` will turn off Langevin dynamics for hydrogen atoms. This parameter has no effect if Langevin coupling coefficients are read from a PDB file.

- `langevinFile` < PDB file containing Langevin parameters >
  Acceptable Values: UNIX filename
  Default Value: `coordinates`
  Description: PDB file to use for the Langevin coupling coefficients for each atom. If this parameter is not specified, then the PDB file specified by `coordinates` is used.

- `langevinCol` < column of PDB from which to read coefficients >
  Acceptable Values: `X`, `Y`, `Z`, `O`, or `B`

Default Value: `O`
Description: Column of the PDB file to use for the Langevin coupling coefficients for each atom. The coefficients can be read from any floating point column of the PDB file. A value of 0 indicates that the atom will remain unaffected.

6.3.2 Temperature coupling parameters

NAMD is capable of performing temperature coupling, in which forces are added or reduced to simulate the coupling of the system to a heat bath of a specified temperature. This capability is based on that implemented in X-PLOR which is detailed in the X-PLOR *User's Manual* [7].

- `tCouple` < perform temperature coupling? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not temperature coupling is active. If set to `on`, then the parameter `tCoupleTemp` must be set and the parameters `tCoupleFile` and `tCoupleCol` can optionally be set to control the behavior of this feature.

- `tCoupleTemp` < temperature for heat bath (K) >
  Acceptable Values: positive decimal
  Description: Temperature to which atoms affected by temperature coupling will be adjusted. This temperature will be roughly maintained across the affected atoms through the addition of forces.

- `tCoupleFile` < PDB file with tCouple parameters >
  Acceptable Values: UNIX filename
  Default Value: `coordinates`
  Description: PDB file to use for the temperature coupling coefficient for each atom. If this parameter is not specified, then the PDB file specified by `coordinates` is used.

- `tCoupleCol` < column of PDB from which to read coefficients >
  Acceptable Values: `X`, `Y`, `Z`, `O`, or `B`
  Default Value: `O`
  Description: Column of the PDB file to use for the temperature coupling coefficient for each atom. This value can be read from any floating point column of the PDB file. A value of 0 indicates that the atom will remain unaffected.

6.3.3 Temperature rescaling parameters

NAMD allows equilibration of a system by means of temperature rescaling. Using this method, all of the velocities in the system are periodically rescaled so that the entire system is set to the desired temperature. The following parameters specify how often and to what temperature this rescaling is performed.

- `rescaleFreq` < number of timesteps between temperature rescaling >
  Acceptable Values: positive integer
  Description: The equilibration feature of NAMD is activated by specifying the number of timesteps between each temperature rescaling. If this value is given, then the `rescaleTemp` parameter must also be given to specify the target temperature.

- `rescaleTemp` < temperature for equilibration (K) >
  Acceptable Values: positive decimal
  Description: The temperature to which all velocities will be rescaled every `rescaleFreq` timesteps. This parameter is valid only if `rescaleFreq` has been set.

6.3.4 Temperature reassignment parameters

NAMD allows equilibration of a system by means of temperature reassignment. Using this method, all of the velocities in the system are periodically reassigned so that the entire system is set to the desired temperature. The following parameters specify how often and to what temperature this reassignment is performed.

- `reassignFreq` < number of timesteps between temperature reassignment >
  Acceptable Values: positive integer
  Description: The equilibration feature of NAMD is activated by specifying the number of timesteps between each temperature reassignment. If this value is given, then the `reassignTemp` parameter must also be given to specify the target temperature.

- `reassignTemp` < temperature for equilibration (K) >
  Acceptable Values: positive decimal
  Default Value: `temperature` if set, otherwise none
  Description: The temperature to which all velocities will be reassigned every `reassignFreq` timesteps. This parameter is valid only if `reassignFreq` has been set.

- `reassignIncr` < temperature increment for equilibration (K) >
  Acceptable Values: decimal
  Default Value: 0
  Description: In order to allow simulated annealing or other slow heating/cooling protocols, `reassignIncr` will be added to `reassignTemp` after each reassignment. (Reassignment is carried out at the first timestep.) The `reassignHold` parameter may be set to limit the final temperature. This parameter is valid only if `reassignFreq` has been set.

- `reassignHold` < holding temperature for equilibration (K) >
  Acceptable Values: positive decimal
  Description: The final temperature for reassignment when `reassignIncr` is set; `reassignTemp` will be held at this value once it has been reached. This parameter is valid only if `reassignIncr` has been set.

6.4 Boundary Conditions

6.4.1 Spherical harmonic boundary conditions

NAMD provides spherical harmonic boundary conditions. These boundary conditions can consist of a single potential or a combination of two potentials. The following parameters are used to define these boundary conditions.

- `sphericalBC` < use spherical boundary conditions? >
  Acceptable Values: `on` or `off`
  Default Value: `off`

Description: Specifies whether or not spherical boundary conditions are to be applied to the system. If set to `on`, then `sphericalBCCenter`, `sphericalBCr1` and `sphericalBCk1` must be defined, and `sphericalBCexp1`, `sphericalBCr2`, `sphericalBCk2`, and `sphericalBCexp2` can optionally be defined.

- `sphericalBCCenter` < center of sphere (Å) >
  Acceptable Values: position
  Description: Location around which sphere is centered.

- `sphericalBCr1` < radius for first boundary condition (Å) >
  Acceptable Values: positive decimal
  Description: Distance at which the first potential of the boundary conditions takes effect. This distance is a radius from the center.

- `sphericalBCk1` < force constant for first potential >
  Acceptable Values: non-zero decimal
  Description: Force constant for the first harmonic potential. A positive value will push atoms toward the center, and a negative value will pull atoms away from the center.

- `sphericalBCexp1` < exponent for first potential >
  Acceptable Values: positive, even integer
  Default Value: 2
  Description: Exponent for first boundary potential. The only likely values to use are 2 and 4.

- `sphericalBCr2` < radius for second boundary condition (Å) >
  Acceptable Values: positive decimal
  Description: Distance at which the second potential of the boundary conditions takes effect. This distance is a radius from the center. If this parameter is defined, then `spericalBCk2` must also be defined.

- `sphericalBCk2` < force constant for second potential >
  Acceptable Values: non-zero decimal
  Description: Force constant for the second harmonic potential. A positive value will push atoms toward the center, and a negative value will pull atoms away from the center.

- `sphericalBCexp2` < exponent for second potential >
  Acceptable Values: positive, even integer
  Default Value: 2
  Description: Exponent for second boundary potential. The only likely values to use are 2 and 4.

6.4.2 Cylindrical harmonic boundary conditions

NAMD provides cylindrical harmonic boundary conditions. These boundary conditions can consist of a single potential or a combination of two potentials. The following parameters are used to define these boundary conditions.

- `cylindricalBC` < use cylindrical boundary conditions? >
  Acceptable Values: `on` or `off`

Default Value: `off`
Description: Specifies whether or not cylindrical boundary conditions are to be applied to the system. If set to `on`, then `cylindricalBCCenter`, `cylindricalBCr1`, `cylindricalBCl1` and `cylindricalBCk1` must be defined, and `cylindricalBCAxis`, `cylindricalBCexp1`, `cylindricalBCr2`, `cylindricalBCl2`, `cylindricalBCk2`, and `cylindricalBCexp2` can optionally be defined.

- `cylindricalBCCenter` < center of cylinder (Å) >
  Acceptable Values: position
  Description: Location around which cylinder is centered.

- `cylindricalBCAxis` < axis of cylinder (Å) >
  Acceptable Values: `x`, `y`, or `z`
  Description: Axis along which cylinder is aligned.

- `cylindricalBCr1` < radius for first boundary condition (Å) >
  Acceptable Values: positive decimal
  Description: Distance at which the first potential of the boundary conditions takes effect along the non-axis plane of the cylinder.

- `cylindricalBCl1` < distance along cylinder axis for first boundary condition (Å) >
  Acceptable Values: positive decimal
  Description: Distance at which the first potential of the boundary conditions takes effect along the cylinder axis.

- `cylindricalBCk1` < force constant for first potential >
  Acceptable Values: non-zero decimal
  Description: Force constant for the first harmonic potential. A positive value will push atoms toward the center, and a negative value will pull atoms away from the center.

- `cylindricalBCexp1` < exponent for first potential >
  Acceptable Values: positive, even integer
  Default Value: 2
  Description: Exponent for first boundary potential. The only likely values to use are 2 and 4.

- `cylindricalBCr2` < radius for second boundary condition (Å) >
  Acceptable Values: positive decimal
  Description: Distance at which the second potential of the boundary conditions takes effect along the non-axis plane of the cylinder. If this parameter is defined, then `cylindricalBCl2` and `spericalBCk2` must also be defined.

- `cylindricalBCl2` < radius for second boundary condition (Å) >
  Acceptable Values: positive decimal
  Description: Distance at which the second potential of the boundary conditions takes effect along the cylinder axis. If this parameter is defined, then `cylindricalBCr2` and `spericalBCk2` must also be defined.

- `cylindricalBCk2` < force constant for second potential >
  Acceptable Values: non-zero decimal
  Description: Force constant for the second harmonic potential. A positive value will push atoms toward the center, and a negative value will pull atoms away from the center.

- `cylindricalBCexp2` < exponent for second potential >
  Acceptable Values: positive, even integer
  Default Value: 2
  Description: Exponent for second boundary potential. The only likely values to use are 2 and 4.

6.4.3 Periodic boundary conditions

NAMD provides periodic boundary conditions in 1, 2 or 3 dimensions. The following parameters are used to define these boundary conditions.

- `cellBasisVector1` < basis vector for periodic boundaries (Å) >
  Acceptable Values: vector
  Default Value: 0 0 0
  Description: Specifies a basis vector for periodic boundary conditions.

- `cellBasisVector2` < basis vector for periodic boundaries (Å) >
  Acceptable Values: vector
  Default Value: 0 0 0
  Description: Specifies a basis vector for periodic boundary conditions.

- `cellBasisVector3` < basis vector for periodic boundaries (Å) >
  Acceptable Values: vector
  Default Value: 0 0 0
  Description: Specifies a basis vector for periodic boundary conditions.

- `cellOrigin` < center of periodic cell (Å) >
  Acceptable Values: position
  Default Value: 0 0 0
  Description: When position rescaling is used to control pressure, this location will remain constant. Also used as the center of the cell for wrapped output coordinates.

- `extendedSystem` < XSC file to read cell parameters from >
  Acceptable Values: file name
  Description: In addition to .coor and .vel output files, NAMD generates a .xsc (eXtended System Configuration) file which contains the periodic cell parameters and extended system variables, such as the strain rate in constant pressure simulations. Periodic cell parameters will be read from this file if this option is present, ignoring the above parameters.

- `XSTfile` < XST file to write cell trajectory to >
  Acceptable Values: file name
  Description: NAMD can also generate a .xst (eXtended System Trajectory) file which contains a record of the periodic cell parameters and extended system variables during the simulation. If `XSTfile` is defined, then `XSTfreq` must also be defined.

- `XSTfreq` < how often to append state to XST file >
  Acceptable Values: positive integer
  Description: Like the `DCDfreq` option, controls how often the extended system configuration will be appended to the XST file.

- `wrapWater` < wrap water coordinates around periodic boundaries? >
  Acceptable Values: on or off
  Default Value: off
  Description: Coordinates are normally output relative to the way they were read in. Hence, if part of a molecule crosses a periodic boundary it is not translated to the other side of the cell on output. This option alters this behavior for water molecules only.

- `wrapAll` < wrap all coordinates around periodic boundaries? >
  Acceptable Values: on or off
  Default Value: off
  Description: Coordinates are normally output relative to the way they were read in. Hence, if part of a molecule crosses a periodic boundary it is not translated to the other side of the cell on output. This option alters this behavior for all contiguous clusters of bonded atoms.

- `wrapNearest` < use nearest image to cell origin when wrapping coordinates? >
  Acceptable Values: on or off
  Default Value: off
  Description: Coordinates are normally wrapped to the diagonal unit cell centered on the origin. This option, combined with `wrapWater` or `wrapAll`, wraps coordinates to the nearest image to the origin, providing hexagonal or other cell shapes.

6.5 Pressure Control

The following options affect all pressure control methods.

- `useGroupPressure` < group or atomic quantities >
  Acceptable Values: `yes` or `no`
  Default Value: `no`
  Description: Pressure can be calculated using either the atomic virial and kinetic energy (the default) or a hydrogen-group based pseudo-molecular virial and kinetic energy. The latter fluctuates less and is required in conjunction with rigidBonds (SHAKE).

- `useFlexibleCell` < anisotropic cell fluctuations >
  Acceptable Values: `yes` or `no`
  Default Value: `no`
  Description: NAMD allows the three orthogonal dimensions of the periodic cell to fluctuate independently when this option is enabled.

- `useConstantRatio` < constant shape in first two cell dimensions >
  Acceptable Values: `yes` or `no`
  Default Value: `no`
  Description: When enabled, NAMD keeps the ratio of the unit cell in the x-y plane constant while allowing fluctuations along all axes. The `useFlexibleCell` option is required for this option.

- `useConstantArea` < constant area and normal pressure conditions >
  Acceptable Values: `yes` or `no`
  Default Value: `no`

Description: When enabled, NAMD keeps the dimension of the unit cell in the x-y plane constant while allowing fluctuations along the z axis. This is not currently implemented in Berendsen's method.

6.5.1 Berendsen pressure bath coupling

NAMD provides constant pressure simulation using Berendsen's method. The following parameters are used to define the algorithm.

- `BerendsenPressure` < use Berendsen pressure bath coupling? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not Berendsen pressure bath coupling is active. If set to `on`, then the parameters `BerendsenPressureTarget`, `BerendsenPressureCompressibility` and `BerendsenPressureRelaxationTime` must be set and the parameter `BerendsenPressureFreq` can optionally be set to control the behavior of this feature.

- `BerendsenPressureTarget` < target pressure (bar) >
  Acceptable Values: positive decimal
  Description: Specifies target pressure for Berendsen's method.

- `BerendsenPressureCompressibility` < compressibility (bar^{-1}) >
  Acceptable Values: positive decimal
  Description: Specifies compressibility for Berendsen's method.

- `BerendsenPressureRelaxationTime` < relaxation time (fs) >
  Acceptable Values: positive decimal
  Description: Specifies relaxation time for Berendsen's method.

- `BerendsenPressureFreq` < how often to rescale positions >
  Acceptable Values: positive multiple of `nonbondedFrequency` and `fullElectFrequency`
  Default Value: `nonbondedFrequency` or `fullElectFrequency` if used
  Description: Specifies number of timesteps between position rescalings for Berendsen's method.

6.5.2 Nosé-Hoover Langevin piston pressure control

NAMD provides constant pressure simulation using a modified Nosé-Hoover method in which Langevin dynamics is used to control fluctuations in the barostat. This method should be combined with a method of temperature control, such as Langevin dynamics, in order to simulate the NPT ensemble. The following parameters are used to define the algorithm.

- `LangevinPiston` < use Langevin piston pressure control? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not Langevin piston pressure control is active. If set to `on`, then the parameters `LangevinPistonTarget`, `LangevinPistonPeriod`, `LangevinPistonDecay` and `LangevinPistonTemp` must be set.

- `LangevinPistonTarget` < target pressure (bar) >
  Acceptable Values: positive decimal
  Description: Specifies target pressure for Langevin piston method.

- `LangevinPistonPeriod` < oscillation period (fs) >
  Acceptable Values: positive decimal
  Description: Specifies barostat oscillation time scale for Langevin piston method.

- `LangevinPistonDecay` < damping time scale (fs) >
  Acceptable Values: positive decimal
  Description: Specifies barostat damping time scale for Langevin piston method.

- `LangevinPistonTemp` < noise temperature (K) >
  Acceptable Values: positive decimal
  Description: Specifies barostat noise temperature for Langevin piston method. This should be set equal to the target temperature for the chosen method of temperature control.

- `SurfaceTensionTarget` < Surface tension target (dyn/cm) >
  Acceptable Values: decimal
  Default Value: 0.0
  Description: Specifies surface tension target. Must be used with `useFlexibleCell` and periodic boundary conditions. The pressure specified in `LangevinPistonTarget` becomes the pressure along the z axis, and surface tension is applied in the x-y plane.

- `StrainRate` < initial strain rate >
  Acceptable Values: decimal triple (x y z)
  Default Value: 0. 0. 0.
  Description: Optionally specifies the initial strain rate for pressure control. Is overridden by value read from file specified with `extendedSystem`.

- `ExcludeFromPressure` < Should some atoms be excluded from pressure rescaling? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not to exclude some atoms from pressure rescaling. The coordinates and velocites of such atoms are not rescaled during constant pressure simulations, though they do contribute to the virial calculation. May be useful for membrane protein simulation. EXPERIMENTAL.

- `ExcludeFromPressureFile` < File specifying excluded atoms >
  Acceptable Values: PDB file
  Default Value: coordinates file
  Description: PDB file with one column specifying which atoms to exclude from pressure rescaling. Specify 1 for excluded and 0 for not excluded.

- `ExcludeFromPressureCol` < Column in PDB file for specifying excluded atoms >
  Acceptable Values: O, B, X, Y, or Z
  Default Value: O
  Description: Specifies which column of the pdb file to check for excluded atoms.

6.6 Applied Forces and Analysis

There are several ways to apply external forces to simulations with NAMD. These are described below.

6.6.1 Constant Forces

NAMD provides the ability to apply constant forces to some atoms. There are two parameters that control this feature.

- `constantforce` < Apply constant forces? >
  Acceptable Values: yes or no
  Default Value: no
  Description: Specifies whether or not constant forces are applied.

- `consforcefile` < PDB file containing forces to be applied >
  Acceptable Values: UNIX filename
  Description: The X, Y, Z and occupancy (O) fields of this file are read to determine the constant force vector of each atom, which is (X,Y,Z)*O, in unit of Kcal/(mol*Å). The occupancy (O) serves as a scaling factor, which could expand the range of the force applied. (One may be unable to record very large or very small numbers in the data fields of a PDB file due to limited space). Zero forces are ignored.

6.6.2 External Electric Field

NAMD provides the ability to apply a constant electric field to the molecular system being simulated. Energy due to the external field will be reported in the MISC column and may be discontinuous in simulations using periodic boundary conditions if, for example, a charged hydrogen group moves outside of the central cell. There are two parameters that control this feature.

- `eFieldOn` < apply electric field? >
  Acceptable Values: `yes` or `no`
  Default Value: `no`
  Description: Specifies whether or not an electric field is applied.

- `eField` < electric field vector >
  Acceptable Values: vector of decimals (x y z)
  Description: Vector which describes the electric field to be applied. Units are kcal/(mol Å e), which is natural for simulations. This parameter may be changed between `run` commands, allowing a square wave or other approximate wave form to be applied.

6.6.3 Moving Constraints

Moving constraints feature works in conjunction with the Harmonic Constraints (see an appropriate section of the User's guide). The reference positions of all constraints will move according to

$$\vec{r}(t) \;=\; \vec{r}_0 + \vec{v}t\,. \qquad (1)$$

A velocity vector $\vec{v}$ (`movingConsVel`) needs to be specified.

The way the moving constraints work is that the moving reference position is calculated every integration time step using Eq. 1, where $\vec{v}$ is in Å/timestep, and t is the current timestep (i.e., `firstTimestep` plus however many timesteps have passed since the beginning of NAMD run). Therefore, one should be careful when restarting simulations to appropriately update the `firstTimestep` parameter in the NAMD configuration file or the reference position specified in the reference PDB file.

NOTE: NAMD actually calculates the constraints potential with $U = k(x - x_0)^d$ and the force with $F = dk(x - x_0)$, where d is the exponent `consexp`. The result is that if one specifies some value for the force constant k in the PDB file, effectively, the force constant is $2k$ in calculations. This caveat was removed in SMD feature.

The following parameters describe the parameters for the moving harmonic constraint feature of NAMD.

- `movingConstraints` < Are moving constraints active >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Should moving restraints be applied to the system. If set to `on`, then `movingConsVel` must be defined. May not be used with `rotConstraints`.

- `movingConsVel` < Velocity of the reference position movement >
  Acceptable Values: vector in Å/timestep
  Description: The velocity of the reference position movement. Gives both absolute value and direction

6.6.4 Rotating Constraints

The constraints parameters are specified in the same manner as for usual (static) harmonic constraints. The reference positions of all constrained atoms are then rotated with a given angular velocity about a given axis. If the force constant of the constraints is sufficiently large, the constrained atoms will follow their reference positions.

A rotation matrix M about the axis unit vector v is calculated every timestep for the angle of rotation corresponding to the current timestep. angle = Ωt, where Ω is the angular velocity of rotation.

From now on, all quantities are 3D vectors, except the matrix M and the force constant K.

The current reference position R is calculated from the initial reference position R_0 (at $t = 0$), $R = M(R_0 - P) + P$, where P is the pivot point.

Coordinates of point N can be found as $N = P + ((R - P) \cdot v)v$. Normal from the atom pos to the axis is, similarly, normal $= (P + ((X - P) \cdot v)v) - X$ The force is, as usual, $F = K(R - X)$; This is the force applied to the atom in NAMD (see below). NAMD does not know anything about the torque applied. However, the torque applied to the atom can be calculated as a vector product torque $= F \times \mathit{normal}$ Finally, the torque applied to the atom with respect to the axis is the projection of the torque on the axis, i.e., $\mathit{torque}_{proj} = \mathit{torque} \cdot v$

If there are atoms that have to be constrained, but not moved, this implementation is not suitable, because it will move *all* reference positions.

Only one of the moving and rotating constraints can be used at a time.

Using very soft springs for rotating constraints leads to the system lagging behind the reference positions, and then the force is applied along a direction different from the "ideal" direction along the circular path.

Pulling on N atoms at the same time with a spring of stiffness K amounts to pulling on the whole system by a spring of stiffness NK, so the overall behavior of the system is as if you are pulling with a very stiff spring if N is large.

In both moving and rotating constraints the force constant that you specify in the constraints pdb file is multiplied by 2 for the force calculation, i.e., if you specified $K = 0.5$ kcal/mol/Å^2 in the pdb file, the force actually calculated is $F = 2K(R - X) = 1$ kcal/mol/Å^2 $(R - X)$. SMD feature of namd2 does the calculation without multiplication of the force constant specified in the config file by 2.

- `rotConstraints` < Are rotating constraints active >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Should rotating restraints be applied to the system. If set to `on`, then `rotConsAxis`, `rotConsPivot` and `rotConsVel` must be defined. May not be used with `movingConstraints`.

- `rotConsAxis` < Axis of rotation >
  Acceptable Values: vector (may be unnormalized)
  Description: Axis of rotation. Can be any vector. It gets normalized before use. If the vector is 0, no rotation will be performed, but the calculations will still be done.

- `rotConsPivot` < Pivot point of rotation >
  Acceptable Values: position in Å
  Description: Pivot point of rotation. The rotation axis vector only gives the direction of the axis. Pivot point places the axis in space, so that the axis goes through the pivot point.

- `rotConsVel` < Angular velocity of rotation >
  Acceptable Values: rate in degrees per timestep
  Description: Angular velocity of rotation, degrees/timestep.

6.6.5 Steered Molecular Dynamics (SMD)

The SMD feature is independent from the harmonic constraints, although it follows the same ideas. In both SMD and harmonic constraints, one specifies a PDB file which indicates which atoms are 'tagged' as constrained. The PDB file also gives initial coordinates for the constraint positions. One also specifies such parameters as the force constant(s) for the constraints, and the velocity with which the constraints move.

There are two major differences between SMD and harmonic constraints:

- In harmonic constraints, each tagged atom is harmonically constrained to a reference point which moves with constant velocity. In SMD, it is the *center of mass* of the tagged atoms which is constrained to move with constant velocity.

- In harmonic constraints, each tagged atom is constrained in all three spatial dimensions. In SMD, tagged atoms are constrained *only along the constraint direction.*

The center of mass of the SMD atoms will be harmonically constrained with force constant k (`SMDk`) to move with velocity v (`SMDVel`) in the direction $\vec{n}$ (`SMDDir`). SMD thus results in the following potential being applied to the system:

$$U(\vec{r}_1, \vec{r}_2, ..., t) \;=\; \frac{1}{2} k \left[vt - (\vec{R}(t) - \vec{R}_0) \cdot \vec{n} \right]^2 . \qquad (2)$$

Here, $t \equiv N_{ts} dt$ where N_{ts} is the number of elapsed timesteps in the simulation and dt is the size of the timestep in femtoseconds. Also, $\vec{R}(t)$ is the current center of mass of the SMD atoms and R_0 is the initial center of mass as defined by the coordinates in SMDFile. Vector $\vec{n}$ is normalized by NAMD before being used.

Output NAMD provides output of the current SMD data. The frequency of output is specified by the SMDOutputFreq parameter in the configuration file. Every SMDOutputFreq timesteps NAMD will print the current timestep, current position of the center of mass of the restrained atoms, and the current force applied to the center of mass (in piconewtons, pN). The output line starts with word SMD

Parameters The following parameters describe the parameters for the SMD feature of NAMD.

- SMD < Are SMD features active >
  Acceptable Values: on or off
  Default Value: off
  Description: Should SMD harmonic constraint be applied to the system. If set to on, then SMDk, SMDFile, SMDVel, and SMDDir must be defined. Specifying SMDOutputFreq is optional.

- SMDFile < SMD constraint reference position >
  Acceptable Values: UNIX filename
  Description: File to use for the initial reference position for the SMD harmonic constraints. All atoms in this PDB file with a nonzero value in the *occupancy* column will be tagged as SMD atoms. The coordinates of the tagged SMD atoms will be used to calculate the initial center of mass. During the simulation, this center of mass will move with velocity SMDVel in the direction SMDDir.

- SMDk < force constant to use in SMD simulation >
  Acceptable Values: positive real
  Description: SMD harmonic constraint force constant. Must be specified in kcal/mol/Å^2. The conversion factor is 1 kcal/mol = 69.479 pN Å.

- SMDVel < Velocity of the SMD reference position movement >
  Acceptable Values: nonzero real, Å/timestep
  Description: The velocity of the SMD center of mass movement. Gives the absolute value.

- SMDDir < Direction of the SMD center of mass movement >
  Acceptable Values: non-zero vector
  Description: The direction of the SMD reference position movement. The vector does not have to be normalized, it is normalized by NAMDbefore being used.

- SMDOutputFreq < frequency of SMD output >
  Acceptable Values: positive integer
  Default Value: 1
  Description: The frequency in timesteps with which the current SMD data values are printed out.

6.6.6 Interactive Molecular Dynamics (IMD)

NAMD now works directly with VMD to allow you to view and interactively steer your simulation. With IMD enabled, you can connect to NAMD at any time during the simulation to view the current state of the system or perform interactive steering.

- `IMDon` < is IMD active? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not to listen for an IMD connection.

- `IMDport` < port number to expect a connection on >
  Acceptable Values: positive integer
  Description: This is a free port number on the machine that node 0 is running on. This number will have to be entered into VMD.

- `IMDfreq` < timesteps between sending coordinates >
  Acceptable Values: positive integer
  Description: This allows coordinates to be sent less often, which may increase NAMD performance or be necessary due to a slow network.

- `IMDwait` < wait for an IMD connection? >
  Acceptable Values: `yes` or `no`
  Default Value: `no`
  Description: If `no`, NAMD will proceed with calculations whether a connection is present or not. If `yes`, NAMD will pause at startup until a connection is made, and pause when the connection is lost.

6.6.7 Tcl interface

NAMD provides a limited Tcl scripting interface designed for applying forces and performing on-the-fly analysis. This interface is efficient if only a few coordinates, either of individual atoms or centers of mass of groups of atoms, are needed. In addition, information must be requested one timestep in advance. The following configuration parameters are used to enable the Tcl interface:

- `tclForces` < is Tcl interface active? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not Tcl interface is active. If it is set to `off`, then no Tcl code is executed. If it is set to `on`, then Tcl code specified in `tclForcesScript` parameters is executed.

- `tclForcesScript` < input for Tcl interface >
  Acceptable Values: file or {script}
  Description: Must contain either the name of a Tcl script file or the script itself between { and } (may include multiple lines). This parameter may occur multiple times and scripts will be executed in order of appearance. The script(s) should perform any required initialization on the Tcl interpreter, including requesting data needed during the first timestep, and define a procedure `calcforces { }` to be called every timestep.

At this point only low-level commands are defined. In the future this list will be expanded. Current commands are:

- `print <anything>`
  This command should be used instead of `puts` to display output. For example, "`print Hello World`".
- `atomid <segname> <resid> <atomname>`
  Determines atomid of an atom from its segment, residue, and name. For example, "`atomid br 2 N`".
- `addatom <atomid>`
  Request coordinates of this atom for next force evaluation. Request remains in effect until `clearconfig` is called. For example, "`addatom 4`" or "`addatom [atomid br 2 N]`".
- `addgroup <atomid list>`
  Request center of mass coordinates of this group for next force evaluation. Returns a group ID which is of the form `gN` where `N` is a small integer. This group ID may then be used to find coordinates and apply forces just like a regular atom ID. Aggregate forces may then be applied to the group as whole. Request remains in effect until `clearconfig` is called. For example, "`set groupid [addgroup { 14 10 12 }]`".
- `clearconfig`
  Clears the current list of requested atoms. After `clearconfig`, calls to `addatom` and `addgroup` can be used to build a new configuration.
- `loadcoords <varname>`
  Loads requested atom and group coordinates (in Å) into a local array. `loadcoords` should only be called from within the `calcforces` procedure. For example, "`loadcoords p`" and "`print p(4)`".
- `loadforces <varname>`
  Loads the forces applied in the previous timestep (in kcal mol^{-1} Å^{-1}) into a local array. `loadforces` should only be called from within the `calcforces` procedure. For example, "`loadforces f`" and "`print f(4)`".
- `loadmasses <varname>`
  Loads requested atom and group masses (in amu) into a local array. `loadmasses` should only be called from within the `calcforces` procedure. For example, "`loadcoords m`" and "`print m(4)`".
- `addforce <atomid|groupid> <force vector>`
  Applies force (in kcal mol^{-1} Å^{-1}) to atom or group. `addforce` should only be called from within the `calcforces` procedure. For example, "`addforce $groupid { 1. 0. 2. }`".

Several vector routines from the VMD Tcl interface are also defined.

6.7 Free Energy of Conformational Change Calculations

NAMD incorporates methods for performing free energy of conformational change perturbation calculations. The system is efficient if only a few coordinates, either of individual atoms or centers of mass of groups of atoms, are needed. The following configuration parameters are used to enable free energy perturbation:

- `freeEnergy` < is free energy perturbation active? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not free energy perturbation is active. If it is set to `off`, then no free energy perturbation is performed. If it is set to `on`, then the free energy perturbation calculation specified in `freeEnergyConfig` parameters is executed.

- `freeEnergyConfig` < free energy perturbation script >
  Acceptable Values: file or {script}
  Description: Must contain either the name of a free energy perturbation script file or the script itself between { and } (may include multiple lines). This parameter may occur multiple times and scripts will be executed in order of appearance. The format of the free energy perturbation script is described below.

The following sections describe the format of the free energy perturbation script.

6.7.1 User-Supplied Conformational Restraints

These restraints extend the scope of the available restraints beyond that provided by the harmonic position restraints. Each restraint is imposed with a potential energy term, whose form depends on the type of the restraint.

Fixed Restraints

Position restraint (1 atom): force constant K_f, and reference position $\overrightarrow{r_{ref}}$

$$E = (K_f/2)\left(|\overrightarrow{r_i} - \overrightarrow{r_{ref}}|\right)^2$$

Stretch restraint (2 atoms): force constant K_f, and reference distance d_{ref}

$$E = (K_f/2)\left(d_i - d_{ref}\right)^2$$

Bend restraint (3 atoms): force constant K_f, and reference angle θ_{ref}

$$E = (K_f/2)\left(\theta_i - \theta_{ref}\right)^2$$

Torsion restraint (4 atoms): energy barrier E_0, and reference angle χ_{ref}

$$E = (E_0/2)\left\{1 - \cos\left(\chi_i - \chi_{ref}\right)\right\}$$

Forcing restraints

Position restraint (1 atom): force constant K_f, and two reference positions $\overrightarrow{r_0}$ and $\overrightarrow{r_1}$

$$E = (K_f/2)\left(|\overrightarrow{r_i} - \overrightarrow{r_{ref}}|\right)^2$$
$$\overrightarrow{r_{ref}} = \lambda\overrightarrow{r_1} + (1-\lambda)\ \overrightarrow{r_0}$$

Stretch restraint (2 atoms): force constant K_f, and two reference distances d_0 and d_1

$$E = (K_f/2)\left(d_i - d_{ref}\right)^2$$
$$d_{ref} = d_1 + (1-\lambda)\,d_0$$

Bend restraint (3 atoms): force constant K_f, and two reference angles θ_0 and θ_1

$$E = (K_f/2)\left(\theta_i - \theta_{ref}\right)^2$$
$$\theta_{ref} = \lambda\theta_1 + (1-\lambda)\,\theta_0$$

Torsion restraint (4 atoms): energy barrier E_0, and two reference angles χ_0 and χ_1

$$E = (E_0/2)\left\{1 - \cos\left(\chi_i - \chi_{ref}\right)\right\}$$
$$\chi_{ref} = \lambda\chi_1 + (1-\lambda)\,\chi_0$$

The forcing restraints depend on the coupling parameter, λ, specified in a conformational forcing calculation. For example, the restraint distance, d_{ref}, depends on λ, and as λ changes two atoms or centers-of-mass are forced closer together or further apart. In this case $K_f = K_{f,0}$, the value supplied at input.

Alternatively, the value of K_f may depend upon the coupling parameter λ according to:
$K_f = K_{f,0}\lambda$

Bounds

Position bound (1 atom): Force constant K_f, reference position $\overrightarrow{r_{ref}}$, and upper or lower reference distance, d_{ref}

Upper bound:

$$E = (K_f/2)\,(d_i - d_{ref})^2 \text{ for } d_i > d_{ref}, \text{ else } E = 0.$$

Lower bound:

$$E = (K_f/2)\,(d_i - d_{ref})^2 \text{ for } d_i < d_{ref}, \text{ else } E = 0.$$

$$d_i^2 = (|\overrightarrow{r_i} - \overrightarrow{r_{ref}}|)^2$$

Distance bound (2 atoms): Force constant K_f, and upper or lower reference distance, d_{ref}

Upper bound:

$$E = (K_f/2)\,(d_{ij} - d_{ref})^2 \text{ for } d_{ij} > d_{ref}, \text{ else } E = 0.$$

Lower bound:

$$E = (K_f/2)\,(d_{ij} - d_{ref})^2 \text{ for } d_{ij} < d_{ref}, \text{ else } E = 0.$$

Angle bound (3 atoms): Force constant K_f, and upper or lower reference angle, θ_{ref}

Upper bound:

$$E = (K_f/2)\,(\theta - \theta_{ref})^2 \text{ for } \theta > \theta_{ref}, \text{ else } E = 0.$$

Lower bound:

$$E = (K_f/2)\,(\theta - \theta_{ref})^2 \text{ for } \theta < \theta_{ref}, \text{ else } E = 0.$$

Torsion bound (4 atoms): An upper and lower bound must be provided together. Energy gap E_0, lower AND upper reference angles, χ_1 and χ_2, and angle interval, $\Delta\chi$.

$$\begin{array}{llll}
\chi_1 & < \chi & < \chi_2 : & E = 0 \\
(\chi_1 - \Delta\chi) & < \chi & < \chi_1 : & E = (G/2)\left\{1 - \cos(\chi - \chi_1)\right\} \\
\chi_2 & < \chi & (\chi_2 + \Delta\chi): & E = (G/2)\left\{1 - \cos(\chi - \chi_2)\right\} \\
(\chi_2 + \Delta\chi) & < \chi & (\chi_1 - \Delta\chi + 2\pi) : & E = G
\end{array}$$

$$G = E_0/\left\{1 - \cos(\Delta\chi)\right\}$$

Bounds may be used in pairs, to set a lower and upper bound. Torsional bounds always are defined in pairs.

6.7.2 Free Energy Calculations

Conformational forcing / Potential of mean force

In conformational forcing calculations, structural parameters such as atomic positions, interatomic distances, and dihedral angles are forced to change by application of changing restraint potentials. For example, the distance between two atoms can be restrained by a potential to a mean distance that is varied during the calculation. The free energy change (or potential of mean force, pmf) for the process can be estimated during the simulation.

The potential is made to depend on a coupling parameter, λ, whose value changes during the simulation. In potential of mean force calculations, the reference value of the restraint potential depends on λ. Alternately, the force constant for the restraint potential may change in proportion to the coupling parameter. Such a calculation gives the value of a restraint free energy, i.e., the free energy change of the system due to imposition of the restraint potential.

Methods for computing the free energy

With conformational forcing (or with molecular transformation calculations) one obtains a free energy difference for a process that is forced on the system by changing the potential energy function that determines the dynamics of the system. One always makes the changing potential depend on a coupling parameter, λ. By convention, λ can have values only in the range from 0 to 1, and a value of $\lambda = 0$ corresponds to one defined state and a value of $\lambda = 1$ corresponds to the other defined state. Intermediate values of λ correspond to intermediate states; in the case of conformational forcing calculations these intermediate states are physically realizable, but in the case of molecular transformation calculations they are not.

The value of λ is changed during the simulation. In the first method provided here, the change in λ is stepwise, while in the second method it is virtually continuous.

Multi-configurational thermodynamic integration (MCTI).

In MCTI one accumulates $\langle \partial U / \partial \lambda \rangle$ at several values of λ, and from these averages estimates the integral

$$-\Delta A = \int \langle \partial U / \partial \lambda \rangle \, d\lambda$$

With this method, the precision of each $\langle \partial U / \partial \lambda \rangle$ can be estimated from the fluctuations of the time series of $\partial U / \partial \lambda$.

Slow growth.

In slow growth, λ is incremented by $\delta\lambda = \pm 1/N_{step}$ after each dynamics integration time-step, and the pmf is estimated as

$$-\Delta A = \Sigma \; (\partial U / \partial \lambda) \; \delta\lambda$$

Typically, slow growth is done in cycles of: equilibration at $\lambda = 0$, change to $\lambda = 1$, equilibration at $\lambda = 1$, change to $\lambda = 0$. It is usual to estimate the precision of slow growth simulations from the results of successive cycles.

6.7.3 Options for Conformational Restraints

User-supplied restraint and bounds specifications

```
urestraint {
  n * (restraint or bound specification)          // see below
}
```

Restraint Specifications (not coupled to pmf calculations)

```
posi    ATOM       kf = KF     ref = (X Y Z)
dist    2 x ATOM   kf = KF     ref = D
angle   3 x ATOM   kf = KF     ref = A
dihe    4 x ATOM   barr = B    ref = A
```

Bound Specifications (not coupled to pmf calculations)

posi bound	ATOM	kf = KF	[low = (X Y Z D) or hi = (X Y Z D)]
dist bound	2 x ATOM	kf = KF	[low = D or hi = D]
angle bound	3 x ATOM	kf = KF	[low = A or hi = A]
dihe bound	4 x ATOM	gap = E	low = A0 hi = A1 delta = A2

Forcing Restraint Specifications (coupled to pmf calculations)

posi pmf	ATOM	kf=KF	low = (X0 Y0 Z0)	hi = (X1 Y1 Z1)
dist pmf	2 x ATOM	kf=KF	low = D0	hi = D1
angle pmf	3 x ATOM	kf=KF	low = A0	hi = A1
dihe pmf	4 x ATOM	barr=B	low = A0	hi = A1

Units

Input item	Units
E, B	kcal/mol
X, Y, Z, D	
A	degrees
K_f	kcal/(mol ^{2}) or kcal/(mol rad^2)

6.7.4 Options for ATOM Specification

The designation ATOM, above, stands for one of the following forms:

A single atom
(segname, resnum, atomname)
Example: (insulin, 10, ca)

All atoms of a single residue
(segname, resnum)
Example: (insulin, 10)

A list of atoms
group { (segname, resnum, atomname), (segname, resnum, atomname), ... }
Example: group { (insulin, 10, ca), (insulin, 10, cb), (insulin, 11, cg) }

All atoms in a list of residues
group { (segname, resnum), (segname, resnum), ... }
Example: group { (insulin, 10), (insulin, 12), (insulin, 14) }

All atoms in a range of residues
group { (segname, resnum) to (segname, resnum) }
Example: group { (insulin, 10) to (insulin, 12) }

One or more atomnames in a list of residues
group { atomname: (segname, resnum), (segname, resnum), ... }
group { (atomname, atomname, ...): (segname, resnum), (segname, resnum), ... }
Examples: group { ca: (insulin, 10), (insulin, 12), (insulin, 14) }
group { (ca, cb, cg): (insulin, 10), (insulin, 12), (insulin, 14) }
group { (ca, cb): (insulin, 10), (insulin, 12) cg: (insulin, 11), (insulin, 12) }

Note: Within a group, atomname is in effect until a new atomname is used, or the keyword all is used. atomname will not carry over from group to group. This note applies to the paragraph below.

One or more atomnames in a range of residues

```
group { atomname: (segname, resnum) to (segname, resnum) }
group { (atomname, atomname, ... ): (segname, resnum) to (segname, resnum) }
```

Examples:

```
group { ca: (insulin, 10) to (insulin, 14) }
group { (ca, cb, cg): (insulin, 10) to (insulin, 12) }
group { (ca, cb): (insulin, 10) to (insulin, 12) all: (insulin, 13) }
```

6.7.5 Options for Potential of Mean Force Calculation

The pmf and mcti blocks, below, are used to simultaneously control all forcing restraints specified in urestraint above. These blocks are performed consecutively, in the order they appear in the config file. The pmf block is used to either a) smoothly vary λ from 0 →1 or 1 →0, or b) set lambda. The mcti block is used to vary λ from 0 →1 or 1 →0 in steps, so that λ is fixed while $dU/d\lambda$ is accumulated.

Lamba control for slow growth

```
pmf {
  task = [up, down, stop, grow, fade, or nogrow]
  time = T [fs, ps, or ns] (default = ps)
  lambda = Y (value of λ; only needed for stop and nogrow)
  lambdat = Z (value of λt; only needed for grow, fade, and nogrow) (default = 0)
  print = P [fs, ps, or ns] or noprint (default = ps)
}
```

up, down, stop:	λ is applied to the reference values.
grow, fade, nogrow:	λ is applied to K_f. A fixed value, λ_t, is used to determine the ref. values.
up, grow:	λ changes from 0 →1. (no value of λ is required)
down, fade:	λ changes from 1 →0. (no value of λ is required)
stop, nogrow:	dU/dλ is accumulated (for single point MCTI)

Lambda control for automated MCTI

```
mcti {
  task = [stepup, stepdown, stepgrow, or stepfade]
  equiltime = T1 [fs, ps, or ns] (default = ps)
  accumtime = T2 [fs, ps, or ns] (default = ps)
  numsteps = N
  lambdat = Z (value of λt; only needed for stepgrow, and stepfade) (default = 0)
  print = P [fs, ps, or ns] or noprint (default = ps)
}
```

stepup, stepdown:	λ is applied to the reference values.
stepgrow, stepfade:	λ is applied to K_f. A fixed value, λ_t, is used to determine the ref. values.
stepup, stepgrow:	λ changes from 0 →1. (no value of λ is required)
stepdown, stepfade:	λ changes from 1 →0. (no value of λ is required)

For each task, λ changes in steps of (1.0/N) from 0 →1 or 1 →0. At each step, no data is accumulated for the initial period T1, then dU/dλ is accumulated for T2. Therefore, the total duration of an mcti block is (T1+T2) x N.

6.7.6 Examples

Fixed restraints

```
    // 1. restrain the position of the ca atom of residue 0.
    // 2. restrain the distance between the ca's of residues 0 and 10 to 5.2Å
    // 3. restrain the angle between the ca's of residues 0-10-20 to 90°.
    // 4. restrain the dihedral angle between the ca's of residues 0-10-20-30 to 180°.
    // 5. restrain the angle between the centers-of-mass of residues 0-10-20 to 90°.
  urestraint {
    posi (insulin, 0, ca) kf=20 ref=(10, 11, 11)
    dist (insulin, 0, ca) (insulin, 10, ca) kf=20 ref=5.2
    angle (insulin, 0, ca) (insulin, 10, ca) (insulin, 20, ca) kf=20 ref=90
    dihe (insulin, 0, ca) (insulin, 10, ca) (insulin, 20, ca) (insulin, 30, ca) barr=20 ref=180
    angle (insulin, 0) (insulin, 10) (insulin, 20) kf=20 ref=90
  }

    // 1.    restrain the center of mass of three atoms of residue 0.
    // 2.    restrain the distance between (the COM of 3 atoms of residue 0) to (the COM of 3 atoms of residue 10).
    // 3.    restrain the dihedral angle of (10,11,12)-(15,16,17,18)-(20,22)-(30,31,32,34,35) to 90°
    //       ( (ca of 10 to 12), (ca, cb, cg of 15 to 18), (all atoms of 20 and 22), (ca of 30, 31, 32, 34, all atoms of 35) ).
  urestraint {
    posi group {(insulin, 0, ca), (insulin, 0, cb), (insulin, 0, cg)} kf=20 ref=(10, 11, 11)
      dist  group {(insulin, 0, ca), (insulin, 0, cb), (insulin, 0, cg)}
            group {(insulin, 10, ca), (insulin, 10, cb), (insulin, 10, cg)} kf=20 ref=6.2
     dihe  group {ca: (insulin, 10) to (insulin, 12)}
           group {(ca, cb, cg): (insulin, 15) to (insulin, 18)}
           group {(insulin, 20), (insulin, 22)}
           group {ca: (insulin, 30) to (insulin, 32), (insulin, 34), all: (insulin, 35)} barr=20 ref=90
  }
```

Bound specifications

```
    // 1.      impose an upper bound if an atom's position strays too far from a reference position.
    //         (add an energy term if the atom is more than 10Å from (2.0, 2.0, 2.0) ).
    // 2&3.    impose lower and upper bounds on the distance between the ca's of residues 5 and 15.
    //         (if the separation is less than 5.0Å or greater than 12.0Å add an energy term).
    // 4.      impose a lower bound on the angle between the centers-of-mass of residues 3-6-9.
    //         (if the angle goes lower than 90° apply a restraining potential).
  urestraint {
    posi bound (insulin, 3, cb) kf=20 hi = (2.0, 2.0, 2.0, 10.0)
    dist bound (insulin, 5, ca) (insulin, 15, ca) kf=20 low = 5.0
    dist bound (insulin, 5, ca) (insulin, 15, ca) kf=20 hi = 12.0
    angle bound (insulin, 3) (insulin, 6) (insulin, 9) kf=20 low=90.0
  }

    // torsional bounds are defined as pairs. this example specifies upper and lower bounds on the
    // dihedral angle, χ, separating the planes of the 1-2-3 residues and the 2-3-4 residues.
    // The energy is 0 for:              -90°   ¡ χ   ¡ 120°
    // The energy is 20 kcal/mol for:    130°   ¡ χ   ¡ 260°
    // Energy rises from 0 → 20 kcal/mol as χ increases from 120° → 130°, and decreases from –90° → –100°.
```

```
urestraint {
  dihe bound (insulin 1) (insulin 2) (insulin 3) (insulin 4) gap=20 low=-90 hi=120 delta=10
}
```

Forcing restraints

```
  // a forcing restraint will be imposed on the distance between the centers-of-mass of residues (10 to 15) and
  // residues (30 to 35). low=20.0, hi=10.0, indicates that the reference distance is 20.0at λ=0, and 10.0at λ=1.
urestraint {
    dist pmf   group { (insulin, 10) to (insulin, 15) }
               group { (insulin, 30) to (insulin, 35) } kf=20, low=20.0, hi=10.0
}
  // 1. during the initial 10 ps, increase the strength of the forcing restraint to full strength: 0 → 20 kcal/(mol ²)
  // 2. next, apply a force to slowly close the distance from 20 to 10 (λ changes from 0 → 1)
  // 3. accumulate dU/dλ for another 10 ps. ( stays fixed at 1)
  // 4. force the distance back to its initial value of 20 ( changes from 1 → 0)
pmf {
  task = grow
  time = 10 ps
  print = 1 ps
}
pmf {
  task = up
  time = 100 ps
}
pmf {
  task = stop
  time = 10 ps
}
pmf {
  task = down
  time = 100 ps
}
  // 1.    force the distance to close from 20 to 10 in 5 steps. (λ changes from 0 →  1:   0.2, 0.4, 0.6, 0.8, 1.0)
  //      at each step equilibrate for 10 ps, then collect dU/dλ for another 10 ps.
  //      ref = 18, 16, 14, 12, 10 , duration = (10 + 10) x 5 = 100 ps.
  // 2.    reverse the step above (λ  changes from 1 → 0:   0.8, 0.6, 0.4, 0.2, 0.0)
mcti {
  task = stepup
  equiltime = 10 ps
  accumtime = 10 ps
  numsteps = 5
  print = 1 ps
}
mcti {
  task = stepdown
}
```

6.7.7 Appendix

Gradient for position restraint

$E = (K_f/2)\left(|\overrightarrow{r_i} - \overrightarrow{r_{ref}}|\right)^2$

$E = (K_f/2)\left\{(x_i - x_{ref})^2 + (y_i - y_{ref})^2 + (z_i - z_{ref})^2\right\}$

$\nabla(E) = K_f\left\{(x_i - x_{ref})\overrightarrow{i} + (y_i - y_{ref})\overrightarrow{j} + (z_i - z_{ref})\overrightarrow{k}\right\}$

Gradient for stretch restraint

$E = (K_f/2)\left(d_i - d_{ref}\right)^2$

$d_i = \left\{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2\right\}^{1/2}$

$\nabla(E) = K_f\left(d_i - d_{ref}\right)\cdot\nabla(di)$

for atom 2 moving, and atom 1 fixed

$\nabla(d_i) = 1/2\left\{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2\right\}^{-1/2}\left\{2(x_2 - x_1) + 2(y_2 - y_1) + 2(z_2 - z_1)\right\}$

$\nabla(d_i) = \left\{(x_2 - x_1)\overrightarrow{i} + (y_2 - y_1)\overrightarrow{j} + (z_2 - z_1)\overrightarrow{k}\right\}/d_i$

$\nabla(E) = K_f\left\{(d_i - d_{ref})/d_i\right\}\left\{(x_2 - x_1)\overrightarrow{i} + (y_2 - y_1)\overrightarrow{j} + (z_2 - z_1)\overrightarrow{k}\right\}$

Gradient for bend restraint

$E = (K_f/2)\left(\theta_i - \theta_{ref}\right)^2$

Atoms at positions A-B-C

distances: (A to B) = c; (A to C) = b; (B to C) = a;

$\theta_i = \cos^{-1}(u) = \cos^{-1}\left\{(a^2 + c^2 - b^2)/(2ac)\right\}$

$\nabla(E) = K_f\left(\theta_i - \theta_{ref}\right)\cdot\nabla(\theta_i)$

$\nabla(\theta_i) = \frac{-1}{\sqrt{1-u^2}}\cdot\nabla(u)$

for atom A moving, atoms B & C fixed (distances b and c change)

$\nabla(u) = \{-b/(ac)\}\cdot\nabla(b) + \left\{-a/(2c^2) + 1/(2a) + b^2/(2ac^2)\right\}\cdot\nabla(c)$

$\nabla(b) = \left\{(x_A - x_C)\overrightarrow{i} + (y_A - y_C)\overrightarrow{j} + (z_A - z_C)\overrightarrow{k}\right\}/b$

$\nabla(c) = \left\{(x_A - x_B)\overrightarrow{i} + (y_A - y_B)\overrightarrow{j} + (z_A - z_B)\overrightarrow{k}\right\}/c$

for atom B moving, atoms A & C fixed (distances a and c change)

$\nabla(u) = \left\{1/(2c) + -c/(2a^2) + b^2/(2a^2c)\right\}\cdot\nabla(a) + \left\{-a/(2c^2) + 1/(2a) + b^2/(2ac^2)\right\}\cdot\nabla(c)$

$\nabla(a) = \left\{(x_B - x_C)\overrightarrow{i} + (y_B - y_C)\overrightarrow{j} + (z_B - z_C)\overrightarrow{k}\right\}/a$

$\nabla(c) = \left\{(x_B - x_A)\overrightarrow{i} + (y_B - y_A)\overrightarrow{j} + (z_B - z_A)\overrightarrow{k}\right\}/c$

for atom C moving, atoms A & B fixed (distances a and b change)

$\nabla(u) = \{-b/(ac)\}\cdot\nabla(b) + \left\{-c/(2a^2) + 1/(2c) + b^2/(2ac^2)\right\}\cdot\nabla(a)$

$\nabla(b) = \left\{(x_C - x_A)\overrightarrow{i} + (y_C - y_A)\overrightarrow{j} + (z_C - z_A)\overrightarrow{k}\right\}/b$

$\nabla(a) = \left\{(x_C - x_B)\overrightarrow{i} + (y_C - y_B)\overrightarrow{j} + (z_C - z_B)\overrightarrow{k}\right\}/a$

Gradient for dihedral angle restraint

$E = (E_0/2)\left\{1 - \cos(\chi_i - \chi_{ref})\right\}$

Atoms at positions A-B-C-D

$\chi_i = \cos^{-1}(u) = \cos^{-1}\left(\frac{(\overrightarrow{CD}\times\overrightarrow{CB})\bullet(\overrightarrow{BC}\times\overrightarrow{BA})}{\left|\overrightarrow{CD}\times\overrightarrow{CB}\right|\left|\overrightarrow{BC}\times\overrightarrow{BA}\right|}\right)$ AND

$\chi_i = \sin^{-1}(v) = \sin^{-1}\left(\frac{(\overrightarrow{CD}\times\overrightarrow{CB})\times(\overrightarrow{BC}\times\overrightarrow{BA})}{\left|\overrightarrow{CD}\times\overrightarrow{CB}\right|\left|\overrightarrow{BC}\times\overrightarrow{BA}\right|}\bullet\frac{\overrightarrow{CB}}{\left|\overrightarrow{CB}\right|}\right)$

$\nabla(E) = (E_0/2)\left\{\sin(\chi_i - \chi_{ref})\right\} \cdot \nabla(\chi_i)$
$\nabla(\chi_i) = \frac{-1}{\sqrt{1-u^2}} \cdot \nabla(u)$

$$\begin{aligned}
\overrightarrow{CD} \times \overrightarrow{CB} = \quad & ((y_D - y_C)(z_B - z_C) - (z_D - z_C)(y_B - y_C))\overrightarrow{i} + \\
& ((z_D - z_C)(x_B - x_C) - (x_D - x_C)(z_B - z_C))\overrightarrow{j} + \\
& ((x_D - x_C)(y_B - y_C) - (y_D - y_C)(x_B - x_C))\overrightarrow{k} \\
= \quad & p_1\overrightarrow{i} + p_2\overrightarrow{j} + p_3\overrightarrow{k}
\end{aligned}$$

$$\begin{aligned}
\overrightarrow{BC} \times \overrightarrow{BA} = \quad & ((y_C - y_B)(z_A - z_B) - (z_C - z_B)(y_A - y_B))\overrightarrow{i} + \\
& ((z_C - z_B)(x_A - x_B) - (x_C - x_B)(z_A - z_B))\overrightarrow{j} + \\
& ((x_C - x_B)(y_A - y_B) - (y_C - y_B)(x_A - x_B))\overrightarrow{k} \\
= \quad & p_4\overrightarrow{i} + p_5\overrightarrow{j} + p_6\overrightarrow{k}
\end{aligned}$$

$u = \frac{p_1p_4+p_2p_5+p_3p_6}{\sqrt{p_1^2+p_2^2+p_3^2}\sqrt{p_4^2+p_5^2+p_6^2}}$

$$\begin{aligned}
\nabla(u) = \quad & \frac{p_1\cdot\nabla(p_4)+p_2\cdot\nabla(p_5)+p_3\cdot\nabla(p_6)}{\sqrt{p_1^2+p_2^2+p_3^2}\sqrt{p_4^2+p_5^2+p_6^2}} + \\
& \frac{p_1p_4+p_2p_5+p_3p_6}{\sqrt{p_1^2+p_2^2+p_3^2}}\left(-1/2\left(p_4^2+p_5^2+p_6^2\right)^{-3/2}\left(2p_4\cdot\nabla(p_4)+2p_5\cdot\nabla(p_5)+2p_6\cdot\nabla(p_6)\right)\right) + \\
& \frac{p_1p_4+p_2p_5+p_3p_6}{\sqrt{p_4^2+p_5^2+p_6^2}}\left(-1/2\left(p_1^2+p_2^2+p_3^2\right)^{-3/2}\left(2p_1\cdot\nabla(p_1)+2p_2\cdot\nabla(p_2)+2p_3\cdot\nabla(p_3)\right)\right)
\end{aligned}$$

for atom A moving, atoms B, C, & D fixed

$$\begin{aligned}
\nabla(p_1) = \quad & (0.0)\overrightarrow{i} + (0.0)\overrightarrow{j} + (0.0)\overrightarrow{k} \\
\nabla(p_2) = \quad & (0.0)\overrightarrow{i} + (0.0)\overrightarrow{j} + (0.0)\overrightarrow{k} \\
\nabla(p_3) = \quad & (0.0)\overrightarrow{i} + (0.0)\overrightarrow{j} + (0.0)\overrightarrow{k} \\
\nabla(p_4) = \quad & (0.0)\overrightarrow{i} + (z_B - z_C)\overrightarrow{j} + (y_C - y_B)\overrightarrow{k} \\
\nabla(p_5) = \quad & (z_C - z_B)\overrightarrow{i} + (0.0)\overrightarrow{j} + (x_B - x_C)\overrightarrow{k} \\
\nabla(p_6) = \quad & (y_B - y_C)\overrightarrow{i} + (x_C - x_B)\overrightarrow{j} + (0.0)\overrightarrow{k}
\end{aligned}$$

for atom B moving, atoms A, C, & D fixed

$$\begin{aligned}
\nabla(p_1) = \quad & (0.0)\overrightarrow{i} + (z_C - z_D)\overrightarrow{j} + (y_D - y_C)\overrightarrow{k} \\
\nabla(p_2) = \quad & (z_D - z_C)\overrightarrow{i} + (0.0)\overrightarrow{j} + (x_C - x_D)\overrightarrow{k} \\
\nabla(p_3) = \quad & (y_C - y_D)\overrightarrow{i} + (x_D - x_C)\overrightarrow{j} + (0.0)\overrightarrow{k} \\
\nabla(p_4) = \quad & (0.0)\overrightarrow{i} + (z_C - z_A)\overrightarrow{j} + (y_A - y_C)\overrightarrow{k} \\
\nabla(p_5) = \quad & (z_A - z_C)\overrightarrow{i} + (0.0)\overrightarrow{j} + (x_C - x_A)\overrightarrow{k} \\
\nabla(p_6) = \quad & (y_C - y_A)\overrightarrow{i} + (x_A - x_C)\overrightarrow{j} + (0.0)\overrightarrow{k}
\end{aligned}$$

Gradient for forcing position restraint

$E = (K_f/2)\left(|\overrightarrow{r_i} - \overrightarrow{r_{ref}}|\right)^2$
$r_{ref} = \lambda\overrightarrow{r_1} + (1-\lambda)\overrightarrow{r_0}$

$$\begin{aligned}
dE/d\lambda = \quad K_f \times \quad & \left((x_i - x_{ref})^2 + (y_i - y_{ref})^2 + (z_i - z_{ref})^2\right)^{1/2} \times \\
& 1/2\left((x_i - x_{ref})^2 + (y_i - y_{ref})^2 + (z_i - z_{ref})^2\right)^{-1/2} \times \\
& \left(2(x_i - x_{ref})(x_0 - x_1) + 2(y_i - y_{ref})(y_0 - y_1) + 2(z_i - z_{ref})(z_0 - z_1)\right)
\end{aligned}$$

$dE/d\lambda = K_f \times \left((x_i - x_{ref})(x_0 - x_1) + (y_i - y_{ref})(y_0 - y_1) + (z_i - z_{ref})(z_0 - z_1)\right)$

Gradient for forcing stretch restraint

$E = (K_f/2)\,(d_i - d_{ref})^2$
$d_{ref} = \lambda d_1 + (1-\lambda)\,d_0$
$dE/d\lambda = K_f \;\times\; (d_i - d_{ref}) \;\times\; (d_0 - d_1)$

Gradient for forcing bend restraint
$E = (K_f/2)\,(\theta_i - \theta_{ref})^2$
$\theta_{ref} = \lambda\theta_1 + (1-\lambda)\,\theta_0$
$dE/d\lambda = K_f \;\times\; (\theta_i - \theta_{ref}) \;\times\; (\theta_0 - \theta_1)$

Gradient for forcing dihedral restraint
$E = (E_0/2)\,(1 - \cos(\chi_i - \chi_{ref}))$
$\chi_{ref} = \lambda\chi_1 + (1-\lambda)\,\chi_0$
$dE/d\lambda = (E_0/2) \;\times\; \sin(\chi_i - \chi_{ref}) \;\times\; (\chi_0 - \chi_1)$

6.8 Alchemical Free Energy Perturbation Calculations

This feature has been contributed to NAMD by the following authors:

Surjit B. Dixit and Christophe Chipot

Equipe de chimie théorique,
Institut nancéien de chimie moléculaire,
UMR CNRS/UHP *7565,*
Université Henri Poincaré,
BP 239,
54506 Vandœuvre–lès–Nancy cedex, France

6.8.1 Introduction and theoretical background

A method to perform alchemical free energy perturbation (FEP) [21, 4, 20, 19, 12, 9, 13, 8] within NAMD has now been implemented. Within FEP, the difference in free energy between two states, a and b, is expressed by:

$$\Delta A_{a\to b} = -k_BT\ \ln\left\langle \exp\left[-\frac{\mathcal{H}_b(\mathbf{r},\mathbf{p}) - \mathcal{H}_a(\mathbf{r},\mathbf{p})}{k_BT}\right]\right\rangle_a \tag{3}$$

wherein k_B is the Boltzmann constant, T is the kinetic temperature, and $\mathcal{H}_a(\mathbf{r},\mathbf{p})$ and $\mathcal{H}_b(\mathbf{r},\mathbf{p})$ are the Hamiltonians characteristic of states a and b, respectively. $\langle\cdots\rangle_a$ denotes an ensemble average over configurations representative of the initial state, a. In practice, the transformation between the two thermodynamic states is replaced by a series of transformations between non–physical, intermediate states along a pathway that connects a to b. This pathway is characterized by a variable, referred to as "coupling parameter", [4, 13, 11] λ, that makes the free energy a continuous function of this parameter between a and b:

$$\Delta A_{a\to b} = -k_BT\ \sum_{k=1}^{N} \ln\left\langle \exp\left[-\frac{\mathcal{H}(\mathbf{r},\mathbf{p};\lambda_{k+1}) - \mathcal{H}(\mathbf{r},\mathbf{p};\lambda_k)}{k_BT}\right]\right\rangle_k \tag{4}$$

Here, N stands for the number of intermediate states, or "windows" between the initial and the final states.

In a typical FEP setup involving the transformation of one chemical species into another one in the course of the simulation, the atoms in the molecular topology can be classified into three groups. A group of atoms that do not change during the simulation — *e.g.*the environment, the atoms describing the initial state, a, of the system, and, last, the atoms that correspond to the final state, b, at the end of the alchemical transformation. The atoms representative of state a do not interact with those of state b throughout the entire molecular dynamics simulation. Such a setup, in which atoms of both the initial and the final states of the system are present in the molecular topology file — *i.e.*the `psf` file — is characterisitic of the so–called "dual topology" protocol. [2] The hybrid Hamiltonian of the system, which is a function of the coupling parameter λ, that smoothly connects state a to state b, is calculated as:

$$\mathcal{H}(\lambda) = \mathcal{H}_0 + \lambda \mathcal{H}_a + (1 - \lambda)\mathcal{H}_b \tag{5}$$

where $\mathcal{H}_a$ is the Hamiltonian of the group of atoms representative of the initial state, a, and $\mathcal{H}_b$ characterizes the final state, b. $\mathcal{H}_0$ is the Hamiltonian for those atoms that do not undergo any transformation during the MD simulation.

In the present implementation of FEP in NAMD, we employ a hamiltonian scaling procedure as is done in the "dual topology" approach, [15] *i.e.*instead of scaling the non–bonded parameters of states a and b, namely the net atomic charges, together with the Lennard–Jones parameters, as a function of the coupling parameter, λ:

$$\begin{aligned} q_i &= \lambda_i \; q_i \\ \epsilon_{ij} &= \sqrt{\lambda_i \; \epsilon_{ii} \; \lambda_j \; \epsilon_{jj}} \\ \sigma_{ij} &= \frac{\lambda_i \; \sigma_{ii} + \lambda_j \; \sigma_{jj}}{2} \end{aligned} \tag{6}$$

where λ_i and λ_j take the value of λ or $(1 - \lambda)$, depending upon state a or b, to which i or j belong.

For instance, in a transformation involving the mutation of an alanine side chain into that of glycine, using the FEP, the topology of both the methyl group borne by the C_α in alanine, and the hydrogen of glycine co–exist throughout the simulation (see Figure 5).

The charge and Lennard–Jones parameters of the alanine and the glycine side chains are defined as a function of λ, in such a fashion that the interaction of the methyl group of alanine with the rest of the protein is effective at the beginning of the simulation, *i.e.*$\lambda = 0$, while the glycine C_α hydrogen does not interact with the rest of the protein, and *vice versa* at the end of the simulation, *i.e.*$\lambda = 1$. For intermediate values of λ, both the alanine and the glycine side chains participate in non–bonded interactions with the rest of the protein, scaled on the basis of the current value of λ. It should be emphasized that these side chains, however, do not interact with themselves. It is not necessary to explicitly exclude those atoms that are created from those that will be annihilated in the course of the FEP calculation.

It is also worth noting that the free energy calculation does not alter intramolecular potentials, *i.e.*bond stretch, valence angle deformation, torsions, *etc.*. . ., during the simulation. In calculations targetted at the estimation of free energy differences between two states characterized by distinct environments — *e.g.*a ligand, bound to a protein in the first simulation, and solvated in water, in

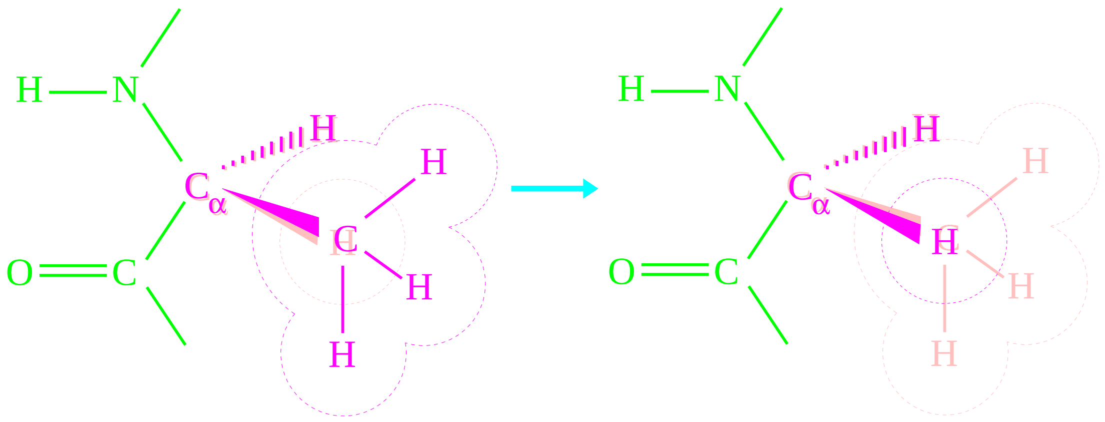

Figure 5: Dual topology description for an alchemical simulation. Case example of the mutation of alanine into glycine. The lighter color denotes the non–interacting, alternate state.

the second — as is the case for most free energy calculations that make use of a thermodynamic cycle, perturbation of intramolecular terms can be safely avoided. [5]

6.8.2 Implementation of free energy perturbation in NAMD

The procedure implemented in NAMD is particularly adapted for performing free energy calculations that split the λ reaction path into a number of non–physical, intermediate states, or "windows". Seperate simulations can be started for each window. Alternatively, the Tcl scripting ability of NAMD can be employed advantageously to perform the complete simulation in a single run. An example making use of such script is supplied at the end of this user guide.

The following keywords can be used to control the alchemical free energy calculations.

- `fep` < Is alchemical FEP to be performed? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Turns on parameter scaling and ensemble averaging for alchemical FEP.

- `lambda` < Coupling parameter value >
  Acceptable Values: positive decimal between 0.0 and 1.0
  Description: The coupling parameter value determining the progress of the perturbation. The non–bonded parameters of the atoms vanishing in the course of the MD simulation are scaled by `lambda`, while the parameters of those atoms grown are scaled by (1-`lambda`).

- `lambda2` < Coupling parameter comparison value >
  Acceptable Values: positive decimal between 0.0 and 1.0
  Description: The `lambda2` value corresponds to the coupling parameter to be used for sampling in the next window. The free energy difference between `lambda2` and `lambda` is calculated. Through simulations at progressive values of `lambda` and `lambda2` the total free energy difference may be determined.

- `fepEquilSteps` < Number of equilibration steps in the window, before data collection >
  Acceptable Values: positive integer less than `numSteps` or `run`

Default Value: 0
Description: In each window `fepEquilSteps` steps of equilibration can be performed before ensemble averaging is initiated. The output also contains the data gathered during equilibration and is meant for analysis of convergence properties of the FEP calculation.

- `fepFile` < pdb file with perturbation flags >
  Acceptable Values: filename
  Default Value: coordinates
  Description: pdb file to be used for indicating the FEP status for each of the atoms pertaining to the system. If this parameter is not declared specifically, then the pdb file containing the initial coordinates specified by `coordinates` is utilized for this information.

- `fepCol` < Column in the `fepFile` that carries the perturbation flag >
  Acceptable Values: X, Y, Z, O or B
  Default Value: B
  Description: Column of the pdb file to use for retrieving the FEP status of each atom, *i.e.*a flag that indicates which atom will be perturbed in the course of the simulation. A value of `-1` in the specified column indicates the atom will vanish during the FEP calculation, whereas a value of `1` indicates that the atom would grow.

- `fepOutFreq` < Frequency of FEP energy output in time–steps >
  Acceptable Values: positive integer
  Default Value: 5
  Description: Every `fepOutFreq` number of MD steps, the output file `fepOutFile` is updated by dumping energies that are used for ensemble averaging. This variable could be set to `1` to include all the configurations for ensemble averaging. Yet, it is recommended to update `fepOutFile` energies with a higher frequency to avoid large correlation between consecutive configurations.

- `fepOutFile` < FEP energy output filename >
  Acceptable Values: filename
  Default Value: outfilename
  Description: An output file named `fepOutFile`.fep, generated by NAMD, contains the FEP energies, dumped every `fepOutFreq` steps.

6.8.3 Examples of input files for running FEP alchemical calculations

The first example illustrates the use of Tcl scripting for running the alchemical FEP feature of NAMD:

```
fep on
fepfile ion.fep
fepCol X
fepOutfile ion.fepout
fepOutFreq 5
fepEquilSteps 5000

set step 0.0
set dstep 0.1
```

```
while {$step <= 0.9} {
 lambda $step
 lambda2 [expr $step+$dstep]
 run  10000
 set step [expr $step+$dstep]
}
```

Here, the pdb file read by NAMD to retrieve the information about perturbed atoms is `biotin.fep`. The pertinent information is present in the `X` column. The output file of the free energy calculation is `biotinr.fepout`, in which energies are written every `5` steps. $\delta\lambda$, the width of the windows, is set to `0.1`. `5000` MD steps are performed in each window to equilibrate the system. In this particular instance, the current value of λ is controlled by the syntax `set step`. The FEP calculation is run until λ reaches the value `0.9`. In every window, `10000` MD steps are being performed.

In the second example, each λ–state is declared explicitly, avoiding the use of Tcl scripting:

```
fep on
fepfile ion.fep
fepCol X
fepOutfile ion.fepout
fepOutFreq 5
fepEquilSteps 5000

lambda 0.0
lambda2 0.1
run 10000

lambda 0.1
lambda2 0.2
run 10000

⋮

lambda 0.8
lambda2 0.9
run 10000

lambda 0.9
lambda2 1.0
run 10000
```

The FEP calculation is carried out from $\lambda =$ `0.0` to `0.9`. In each new window, `10000` MD steps are performed.

6.8.4 Description of FEP simulation output

The `fepOutFile` contains electrostatic and van der Waals energy data calculated for λ and $lambda2$, written every `fepOutFreq` steps. The column `dE` is the energy difference of the single configuration, `dE_avg` and `dG` are the instantaneous ensemble average of the energy and the calculated free energy at the time step specified in column 2, respectively. The temperature is specified in the penultimate column. On completion of `fepEquilSteps` steps, the calculation of `dE_avg` and `dG` is restarted. The accumulated net free energy change is output and the end of the simulation at each lambda value. The cummulative average energy `dE_avg` value may be summed using the trapezoidal rule to obtain an approximate TI estimate for the free energy change during the run.

6.9 Locally Enhanced Sampling

Locally enhanced sampling (LES) [16, 17, 18] increases sampling and transition rates for a portion of a molecule by the use of multiple non-interacting copies of the enhanced atoms. These enhanced atoms experience a nonbonded (electrostatics and van der Waals) potential that is divided by the number of copies present; the bonded potential is not affected. In this way the enhanced atoms can occupy the same space, while the multiple instances and reduces barriers imposed by the nonbonded interactions increase transition rates.

6.9.1 Structure Generation

To use LES, the structure and coordinate input files must be modified to contain multiple copies of the enhanced atoms. `psfgen` provides the `multiply` command for this purpose. NAMD supports a maximum of 15 copies, which should be sufficient. You may create more copies than you intend to use in a every simulation and NAMD will scale the nonbonded potential of these atoms to zero.

Begin by generating the complete molecular structure and guessing coordinates as described in Sec. 4. As the last operation in your script, prior to writing the psf and pdb files, add the `multiply` command, specifying the number of copies desired and listing segments, residues, or atoms to be multiplied. For example, `multiply 4 BPTI:56 BPTI:57` will create four copies of the last two residues of segment BPTI. You must include all atoms to be enhanced in a single `multiply` command in order for the bonded terms in the psf file to be duplicated correctly. Calling `multiply` on connected sets of atoms multiple times will produce unpredictable results, as may running other commands after `multiply`.

The enhanced atoms are duplicated exactly in the structure—they have the same segment, residue, and atom names. They are distinguished only by the value of the B (beta) column in the pdb file, which is 0 for normal atoms and varies from 1 to the number of copies created for enhanced atoms. The enhanced atoms may be easily observed in VMD with the atom selection `beta != 0`.

6.9.2 Simulation

In practice, LES is a simple method used to increase sampling; no special output is generated. The following parameters are used to enable LES:

- `les` < is locally enhanced sampling active? >
  Acceptable Values: `on` or `off`
  Default Value: `off`
  Description: Specifies whether or not LES is active.

- `lesFactor` < number of LES images to use >
  Acceptable Values: positive integer equal to or less than the number of images present
  Description: This should generally be equal to the factor used in `multiply` when creating

the structure. The nonbonded potential of images is divided by `lesFactor`. If atoms are present with an `lesCol` value in `lesFile` that is smaller than `lesFactor`, then the nonbonded potential of these atoms is set to zero.

- `lesFile` < PDB file containing LES flags >
  Acceptable Values: UNIX filename
  Default Value: `coordinates`
  Description: PDB file to specify the LES image number of each atom. If this parameter is not specified, then the PDB file containing initial coordinates specified by `coordinates` is used.

- `lesCol` < column of PDB file containing LES flags >
  Acceptable Values: `X`, `Y`, `Z`, `O`, or `B`
  Default Value: `B`
  Description: Column of the PDB file to specify the LES image number of each atom. This parameter may specify any of the floating point fields of the PDB file, either X, Y, Z, occupancy, or beta-coupling (temperature-coupling). A value of 0 in this column indicates that the atom is not enhanced. Any other value should be a positive integer less than 16.

The parameter `lesFactor` may be varied between simulations to interpolate between full enhancement and normal simulation (although multiple bonded images are present at all times). If `lesFactor` is decreased, the images with flags greater than `lesFactor` will be decoupled from nonbonded terms, sample based only on bonded terms, and should therefore be excluded from analysis. When increasing `lesFactor` the coordinates of these abandoned images should be reset to that of another image to avoid any bad initial contacts and the resulting instability in the simulation.

7 Translation between NAMD and X-PLOR configuration parameters

NAMD was designed to provide many of the same molecular dynamics functions that X-PLOR provides. As such, there are many similarities between the types of parameters that must be passed to both X-PLOR and NAMD. This section describes relations between similar NAMD and X-PLOR parameters.

- **NAMD Parameter:** `cutoff`
  X-PLOR Parameter: `CTOFNB`
  When full electrostatics are not in use within NAMD, these parameters have exactly the same meaning — the distance at which electrostatic and van der Waals forces are truncated. When full electrostatics are in use within NAMD, the meaning is still very similar. The van der Waals force is still truncated at the specified distance, and the electrostatic force is still computed at every timestep for interactions within the specified distance. However, the NAMD integration uses multiple time stepping to compute electrostatic force interactions beyond this distance every `stepspercycle` timesteps.
- **NAMD Parameter:** `vdwswitchdist`
  X-PLOR Parameter: `CTONNB`
  Distance at which the van der Waals switching function becomes active.
- **NAMD Parameter:** `pairlistdist`
  X-PLOR Parameter: `CUTNb`
  Distance within which interaction pairs will be included in pairlist.
- **NAMD Parameter:** `1-4scaling`
  X-PLOR Parameter: `E14Fac`
  Scaling factor for 1-4 pair electrostatic interactions.
- **NAMD Parameter:** `dielectric`
  X-PLOR Parameter: `EPS`
  Dielectric constant.
- **NAMD Parameter:** `exclude`
  X-PLOR Parameter: `NBXMod`
  Both parameters specify which atom pairs to exclude from non-bonded interactions. The ability to ignore explicit exclusions is *not* present within NAMD, thus only positive values of `NBXMod` have NAMD equivalents. These equivalences are
  - `NBXMod=1` is equivalent to `exclude=none` — no atom pairs excluded,
  - `NBXMod=2` is equivalent to `exclude=1-2` — only 1-2 pairs excluded,
  - `NBXMod=3` is equivalent to `exclude=1-3` — 1-2 and 1-3 pairs excluded,
  - `NBXMod=4` is equivalent to `exclude=1-4` — 1-2, 1-3, and 1-4 pairs excluded,
  - `NBXMod`=5 is equivalent to `exclude=scaled1-4` — 1-2 and 1-3 pairs excluded, 1-4 pairs modified.

- **NAMD Parameter:** `switching`
  X-PLOR Parameter: `SHIFt, VSWItch, and TRUNcation`
  Activating the NAMD option `switching` is equivalent to using the X-PLOR options `SHIFt` and `VSWItch`. Deactivating `switching` is equivalent to using the X-PLOR option `TRUNcation`.

- **NAMD Parameter:** `temperature`
  X-PLOR Parameter: `FIRSttemp`
  Initial temperature for the system.

- **NAMD Parameter:** `rescaleFreq`
  X-PLOR Parameter: `IEQFrq`
  Number of timesteps between velocity rescaling.

- **NAMD Parameter:** `rescaleTemp`
  X-PLOR Parameter: `FINAltemp`
  Temperature to which velocities are rescaled.

- **NAMD Parameter:** `restartname`
  X-PLOR Parameter: `SAVE`
  Filename prefix for the restart files.

- **NAMD Parameter:** `restartfreq`
  X-PLOR Parameter: `ISVFrq`
  Number of timesteps between the generation of restart files.

- **NAMD Parameter:** `DCDfile`
  X-PLOR Parameter: `TRAJectory`
  Filename for the position trajectory file.

- **NAMD Parameter:** `DCDfreq`
  X-PLOR Parameter: `NSAVC`
  Number of timesteps between writing coordinates to the trajectory file.

- **NAMD Parameter:** `velDCDfile`
  X-PLOR Parameter: `VELOcity`
  Filename for the velocity trajectory file.

- **NAMD Parameter:** `velDCDfreq`
  X-PLOR Parameter: `NSAVV`
  Number of timesteps between writing velocities to the trajectory file.

- **NAMD Parameter:** `numsteps`
  X-PLOR Parameter: `NSTEp`
  Number of simulation timesteps to perform.

8 Sample configuration files

This section contains some simple example NAMD configuration files to serve as templates.

This file shows a simple configuration file for alanin. It performs basic dynamics with no output files or special features.

```
# protocol params
numsteps        1000

# initial config
coordinates     alanin.pdb
temperature     300K
seed            12345

# output params
outputname      /tmp/alanin
binaryoutput    no

# integrator params
timestep        1.0

# force field params
structure       alanin.psf
parameters      alanin.params
exclude         scaled1-4
1-4scaling      1.0
switching       on
switchdist      8.0
cutoff          12.0
pairlistdist    13.5
stepspercycle   20
```

This file is again for alanin, but shows a slightly more complicated configuration. The system is periodic, a coordinate trajectory file and a set of restart files are produced.

```
# protocol params
numsteps        1000

# initial config
coordinates     alanin.pdb
temperature     300K
seed            12345

# periodic cell
cellBasisVector1   33.0 0 0
cellBasisVector2   0 32.0 0
cellBasisVector3   0 0 32.5

# output params
outputname      /tmp/alanin
binaryoutput    no
DCDfreq         10
restartfreq     100

# integrator params
timestep        1.0

# force field params
structure       alanin.psf
parameters      alanin.params
exclude         scaled1-4
1-4scaling      1.0
switching       on
switchdist      8.0
cutoff          12.0
pairlistdist    13.5
stepspercycle   20
```

This file shows another simple configuration file for alanin, but this time with full electrostatics using PME and multiple timestepping.

```
# protocol params
numsteps        1000

# initial config
coordinates     alanin.pdb
temperature     300K
seed            12345

# periodic cell
cellBasisVector1   33.0 0 0
cellBasisVector2   0 32.0 0
cellBasisVector3   0 0 32.5

# output params
outputname      /tmp/alanin
binaryoutput    no
DCDfreq         10
restartfreq     100

# integrator params
timestep        1.0
fullElectFrequency  4

# force field params
structure       alanin.psf
parameters      alanin.params
exclude         scaled1-4
1-4scaling      1.0
switching       on
switchdist      8.0
cutoff          12.0
pairlistdist    13.5
stepspercycle   20

# full electrostatics
PME             on
PMEGridSizeX    32
PMEGridSizeY    32
PMEGridSizeZ    32
```

9 Running NAMD

NAMD runs on a variety of platforms. Details of running on each specific platform are given below and in the release notes included in every distribution.

9.1 Individual Workstations

Individual workstations use the same version of NAMD as workstation networks, but running NAMD is much easier. You may launch any number of namd2 processes on the local machine (for best performance lauch one process per processor) using the `++local` option via:

```
  charmrun namd2 ++local +p<procs> <configfile>
```

There is no longer any need to be able to `rsh localhost` or to create a nodelist file containing the single host localhost.

Intel and Alpha processors produce binary files (restart and DCD files) which must be `byte-swapped` to be read on other platforms. NAMD and VMD now handle this conversion automatically for most files.

9.2 Individual Windows Workstations

NAMD may be run on a single Windows workstation via the command:

```
  charmrun namd2 ++local +p<procs> <configfile>
```

For best performance, ¡procs¿ should be the number of processors in your machine, and defaults to one if the `+p` option is omitted. However, the `++local` option is required unless `charmd` is running and a nodelist file (containing only localhost) is present.

See below to run on multiple machines.

9.3 Workstation Networks

Workstation networks require two files, the namd2 executable and the charmrun program. The charmrun program starts namd2 on the desired hosts, and handles console I/O for the node programs.

To specify what machines namd2 will run on, the user creates a file called `nodelist`. Below is an example nodelist file:

```
group main
 host brutus
 host romeo
```

The `group main` line defines the default machine list. Hosts brutus and romeo are the two machines on which to run the simulation. Note that charmrun may run on one of those machines, or charmrun may run on a third machine.

The `rsh` command (`remsh` on HPUX) is used to start namd2 on each node specified in the nodelist file. If NAMD fails without printing any output, check to make sure that `rsh` works on your machine, by seeing if `rsh hostname ls` works for each host in the nodelist. If you want or need to use `ssh` instead, then add `setenv CONV_RSH ssh` to your login or batch script and try `ssh hostname ls` to each host first to ensure that the machine is in your `.ssh/known_hosts` file. If you are unable to use rsh or ssh, then add `setenv CONV_DAEMON` and run `charmd` (or `charmd_faceless`, which produces a log file) on every node.

Some automounters use a temporary mount directory which is prepended to the path returned by the pwd command. To run on multiple machines you must add a `++pathfix` option to your nodelist file. For example:

```
group main ++pathfix /tmp_mnt /
 host alpha1
 host alpha2
```

A number of parameters may be passed to charmrun. The most important is the `+pX` option, where X specifies the number of processors. If X is less than the number of hosts in the nodelist, machines are selected from top to bottom. If X is greater than the number of hosts, charmrun will start multiple processes on the machines, starting from the top. To run multiple processes on members of a SMP workstation cluster, you may either just use the +p option to go through the list the right number of times, or list each machine several times, once for each processor. The default is +p1.

You may specify the nodelist file with the `++nodelist` option and the group (which defaults to `main`) with the `++nodegroup` option. If you do not use `++nodelist` charmrun will first look for `nodelist` in your current directory and then `.nodelist` in your home directory.

If you always want to run on the machine you are logged in to you may use `localhost` in place of the hostname in your nodelist file, but only if there are no other machines. You will not need `++pathfix`. For example, `.nodelist` in your home directory could read:

```
group main
 host localhost
```

It is simpler in many cases to instead use the `++local` option as described under `Individual Workstations` above, which eliminates the need for the nodelist file and rsh entirely.

Once the nodelist file is set up, and you have your configuration file prepared, run NAMD as follows:

```
  charmrun +p<procs> namd2 <configfile>
```

9.4 Windows Workstation Networks

Windows is the same as other workstation networks described above, except that rsh is not available on this platform. Instead, you must run the provided daemon (`charmd.exe`) on every node listed in the nodelist file. Using `charmd_faceless` rather than `charmd` will eliminate consoles for the daemon and node processes. The `++local` option is also available under Windows, eliminating the need for `charmd` and nodelist when running NAMD only on the local machine.

9.5 Scyld Beowulf Clusters

Scyld Beowulf clusters replace rsh and other methods of launching jobs via a distributed process space. There is no need for a nodelist file or any special daemons. In order to allow access to files, the first NAMD process must be on the master node of the cluster. Launch jobs from the master node of the cluster via the command:

```
charmrun namd2 +p<procs> <configfile>
```

For best performance, run a single NAMD job on all available nodes and never run multiple NAMD jobs at the same time. You may safely suspend and resume a running NAMD job on a Scyld Beowulf using control-Z or `kill -STOP` and `kill -CONT` on the charmrun process.

9.6 Compaq AlphaServer SC

Although NAMD uses MPI and the Elan library on this platform, parallel jobs are run using the `prun` command. (The standard MPI charmrun is wrong on this platform.) The syntax for this command is:

```
prun -n <procs> <configfile>
```

There are additional options. Consult your local documentation.

9.7 IBM RS/6000 SP

Run NAMD as you would any POE program. The options and environment variables for poe are various and arcane, so you should consult your local documentation for recommended settings. As an example, to run on Blue Horizon one would specify:

```
poe namd2 <configfile> -nodes <procs/8> -tasks_per_node 8
```

9.8 Cray T3E

The T3E version has been tested on the Pittsburgh Supercomputer Center T3E. To run on ¡procs¿ processors, use the mpprun command:

```
mpprun -n <procs> namd2 <configfile>
```

9.9 Origin 2000

For small numbers of processors (1-8) use the non-MPI version of namd2. If your stack size limit is unlimited, which DQS may do, you will need to set it with `limit stacksize 64M` to run on multiple processors. To run on ¡procs¿ processors call the binary directly with the +p option:

```
namd2 +p<procs> <configfile>
```

For better performance on larger numbers of processors we recommend that you use the MPI version of NAMD. To run this version, you must have MPI installed. Furthermore, you must set two environment variables to tell MPI how to allocate certain internal buffers. Put the following commands in your .cshrc or .profile file, or in your job file if you are running under a queuing system:

```
setenv MPI_REQUEST_MAX 10240
setenv MPI_TYPE_MAX 10240
```

Then run NAMD with the following command:

```
mpirun -np <procs> namd2 <configfile>
```

10 NAMD Availability and Installation

NAMD is distributed freely for non-profit use. NAMD 2.5b1 is based on the Charm messaging system and the Converse communication layer (`http://charm.cs.uiuc.edu/`) which have been ported to a wide variety of parallel platforms. This section describes how to obtain and install NAMD 2.5b1.

10.1 How to obtain NAMD

NAMD may be downloaded from `http://www.ks.uiuc.edu/Research/namd/`. You will be required to provide minimal registration information and agree to a license before receiving access to the software. Both source and binary distributions are available.

10.2 Platforms on which NAMD will currently run

NAMD should be portable to any parallel platform with a modern C++ compiler to which Charm and Converse have been ported. Precompiled NAMD 2.5b1 binaries are available for the following platforms:

- Windows (NT, etc.) on Intel processors
- Mac OS X (also called Darwin) on PowerPC processors
- AIX on RS/6000 processors
- HP-UX on PA-RISC processors
- Linux on Intel and Alpha processors
- Scyld Beowulf on Intel processors
- Solaris on Sparc processors (with and without MPI)
- Tru64 Unix on Alpha processors
- Cray T3E
- IBM RS/6000 SP
- Compaq AlphaServer SC (with MPI and Elan)
- SGI Origin 2000 (with and without MPI)

10.3 Compiling NAMD

We provide complete and optimized binaries for all platforms to which NAMD has been ported. It should not be necessary for you to compile NAMD unless you wish to add or modify features.

Directions for compiling NAMD are contained in the release notes, which are available from the NAMD web site `http://www.ks.uiuc.edu/Research/namd/` and are included in all distributions.

10.4 Documentation

All available NAMD documentation is available for download without registration via the NAMD web site `http://www.ks.uiuc.edu/Research/namd/`.

References

[1] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids.* Oxford University Press, New York, 1987.

[2] P. H. Axelsen and D. Li. Improved convergence in dual-topology free energy calculations through use of harmonic restraints. *J. Comput. Chem.*, 19:1278–1283, 1998.

[3] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, J. E. F. Meyer, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank: A computer-based archival file for macromolecular structures. *J. Mol. Biol.*, 112:535–542, 1977.

[4] D. L. Beveridge and F. M. DiCapua. Free energy via molecular simulation: applications to chemical and biomolecualr systems. *Ann. Rev. Biophys. Biophys. Chem.*, 18:431–492, 1989.

[5] S. Boresch and M. Karplus. The role of bonded terms in free energy simulations. 1. theoretical analysis. *J. Phys. Chem. A*, 103:103–118, 1999.

[6] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. CHARMM: a program for macromolecular energy, minimization, and dynamics calculations. *J. Comp. Chem.*, 4(2):187–217, 1983.

[7] A. T. Brünger. *X-PLOR, Version 3.1, A System for X-ray Crystallography and NMR.* The Howard Hughes Medical Institute and Department of Molecular Biophysics and Biochemistry, Yale University, 1992.

[8] C. Chipot and D. A. Pearlman. Free energy calculations. the long and winding gilded road. *Mol. Sim.*, 2001.

[9] M. K. Gilson, J. A. Given, B. L. Bush, and J. A. McCammon. The statistical-thermodynamic basis for computation of binding affinities: a critical review. *Biophys. J.*, 72:1047–1069, 1997.

[10] W. Humphrey and A. Dalke. VMD user guide (Version 0.94). Beckman Institute Technical Report TB-94-07, University of Illinois, 1994.

[11] P. M. King. Free energy via molecular simulation: A primer. In W. F. van Gunsteren, P. K. Weiner, and A. J. Wilkinson, editors, *Computer simulation of biomolecular systems: Theoretical and experimental applications*, volume 2, pages 267–314. ESCOM, Leiden, 1993.

[12] P. A. Kollman. Free energy calculations: applications to chemical and biochemical phenomena. *Chem. Rev.*, 93:2395–2417, 1993.

[13] A. E. Mark. Free energy perturbation calculations. In P. Schleyer, editor, *Encyclopaedia of computational chemistry*, volume 2, pages 1070–1083. John Wiley and Sons, New York, 1998.

[14] J. A. McCammon and S. C. Harvey. *Dynamics of Proteins and Nucleic Acids.* Cambridge University Press, Cambridge, 1987.

[15] D. A. Pearlman. A comparison of alternative approaches to free energy calculations. *J. Phys. Chem.*, 98:1487–1493, 1993.

[16] A. Roitberg and R. Elber. Modeling side chains in peptides and proteins: Application of the locally enhanced sampling technique and the simulated annealing methods to find minimum energy conformations. 95:9277–9287, 1991.

[17] C. Simmerling, T. Fox, and P. A. Kollman. Use of locally enhanced sampling in free energy calculations: Testing and application to the $\alpha \rightarrow \beta$ anomerization of glucose. 120(23):5771–5782, 1998.

[18] C. Simmerling, M. R. Lee, A. R. Ortiz, A. Kolinski, J. Skolnick, and P. A. Kollman. Combining MONSSTER and LES/PME to predict protein structure from amino acid sequence: Application to the small protein CMTI-1. 122(35):8392–8402, 2000.

[19] T. Straatsma and A. McCammon. Computational alchemy. *Ann. Rev. Phys. Chem.*, 43:407–435, 1992.

[20] W. F. van Gunsteren. Methods for calculation of free energies and binding constants: successes and problems. In W. F. Van Gunsteren and P. K. Weiner, editors, *Computer simulation of biomolecular systems: theoretical and experimental applications*, pages 27–59. ESCOM Science Publishers B. V., The Netherlands, 1989.

[21] R. W. Zwanzig. High temperature equation of state by a perturbation method. i. nonpolar gases. *J. Chem. Phys.*, 22:1420–1426, 1954.