NAMD User's Guide

Version 2.0

M. Bhandarkar, R. Brunner, A. Dalke, A. Gursoy, W. Humphrey,
D. Hurwitz, N. Krawetz, M. Nelson, J. Phillips, A. Shinozaki

March 22, 1999

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 2.0

Authors: M. Bhandarkar, R. Brunner, A. Dalke, A. Gursoy, W. Humphrey, D. Hurwitz, N. Krawetz, M. Nelson, J. Phillips, A. Shinozaki

Theoretical Biophysics Group, Beckman Institute, University of Illinois.

NOTICE

Contents

1 Introduction **6**
1.1 New features in version 2.0 . . . 6
1.2 NAMD and molecular dynamics simulations . . . 7
1.3 User feedback . . . 9
1.4 Acknowledgments . . . 11

2 Getting Started **12**
2.1 What is needed . . . 12
2.2 NAMD configuration file . . . 12
2.2.1 Configuration parameter syntax . . . 12
2.2.2 Required NAMD configuration parameters . . . 13

3 Input and Output Files **14**
3.1 File formats . . . 14
3.1.1 PDB files . . . 14
3.1.2 X-PLOR format PSF files . . . 14
3.1.3 CHARMM19 and CHARMM22 parameter files . . . 14
3.1.4 DCD trajectory files . . . 14
3.2 NAMD configuration parameters . . . 14
Input files . . . 14
Output files . . . 16

4 Basic Simulation Parameters **19**
4.1 Non-bonded interaction parameters and computations . . . 19
4.1.1 Non-bonded van der Waals interactions . . . 19
4.1.2 Non-bonded electrostatic interactions . . . 19
4.1.3 Nonbonded interaction distance-testing . . . 21
4.2 Full electrostatic integration . . . 22
4.3 NAMD configuration parameters . . . 23
4.3.1 Timestep parameters . . . 23
4.3.2 Simulation space partitioning . . . 23
4.3.3 Basic dynamics . . . 24
4.3.4 DPMTA parameters . . . 26
4.3.5 DPME parameters . . . 27
4.3.6 Full direct parameters . . . 28
4.3.7 Multiple timestep parameters . . . 28

5 Additional Simulation Parameters **30**
5.1 Constraints and Restraints . . . 30
5.1.1 Harmonic constraint parameters . . . 30
5.1.2 Fixed atoms parameters . . . 31
5.2 Energy Minimization . . . 32
5.2.1 Velocity quenching parameters . . . 32
5.3 Temperature Control and Equilibration . . . 32
5.3.1 Langevin dynamics parameters . . . 32

5.3.2 Temperature coupling parameters . . . 33
5.3.3 Temperature rescaling parameters . . . 34
5.3.4 Temperature reassignment parameters . . . 34
5.4 Boundary Conditions . . . 35
5.4.1 Spherical harmonic boundary conditions . . . 35
5.4.2 Cylindrical harmonic boundary conditions . . . 36
5.4.3 Periodic boundary conditions . . . 37
5.5 Pressure Control . . . 38
5.5.1 Berendsen pressure bath coupling . . . 39
5.5.2 Nosé-Hoover Langevin piston pressure control . . . 39
5.6 Applied Forces and Analysis . . . 40
5.6.1 Moving Constraints . . . 40
5.6.2 Rotating Constraints . . . 41
5.6.3 Steered Molecular Dynamics (SMD) . . . 42
5.6.4 Tcl interface . . . 46
5.7 Free Energy of Conformational Change Calculations . . . 48
5.7.1 User-Supplied Conformational Restraints . . . 48
5.7.2 Free Energy Calculations . . . 50
5.7.3 Options for Conformational Restraints . . . 50
5.7.4 Options for ATOM Specification . . . 51
5.7.5 Options for Potential of Mean Force Calculation . . . 52
5.7.6 Examples . . . 53
5.7.7 Appendix . . . 55

6 Translation between NAMD and X-PLOR configuration parameters 58

7 Sample configuration files 60

8 Running NAMD 63
8.1 Platform-Specific Notes . . . 63
8.1.1 Network of Workstations . . . 63
8.1.2 IBM SP3 . . . 63
8.1.3 CRAY T3E . . . 63
8.1.4 Origin2000 . . . 64
8.2 Interactive modeling with MDScope . . . 64

9 NAMD Availability and Installation 65
9.1 How to obtain NAMD . . . 65
9.2 Platforms on which NAMD will currently run . . . 65
9.3 Compiling NAMD . . . 65
9.4 Documentation . . . 66

List of Figures

1 Graph of van der Waals potential with and without switching 19
2 Graph of electrostatic potential with and without shifting function 20
3 Graph of electrostatic split between short and long range forces 20
4 Example of cutoff and pairlist distance uses 22

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 lists basic simulation options.

Section 5 lists additional simulation options.

Section 6 provides hints for X-PLOR users.

Section 7 provides sample configuration files.

Section 8 gives details on running NAMD.

Section 9 gives details on installing NAMD.

We have attempted to make this document and the NAMD *Programmer's Guide* both complete and easy to understand and to make NAMD itself easy to install and run. Please take a moment to help us improve the documentation and the code by filling out the users' survey at the end of this section.

1.1 New features in version 2.0

There are many structural differences between versions 1.X and 2.X of NAMD. Major portions of the code have been completely rewritten, resulting in a program which is significantly faster and uses less memory than earlier versions. This new architecture also provides support for more sophisticated algorithms and is the base for future efficient support of clusters of symmetric multiprocessor workstations. The most noticeable change to the end user is the switch from PVM to the Converse parallel message-passing system. NAMD 2.0 has also been ported to several new platforms.

New features in NAMD 2.0 include periodic boundary conditions, particle-mesh Ewald electrostatic evaluation, Berendsen and Langevin Nosé-Hoover constant pressure simulation methods, fixed atoms, several boundary conditions, triple timestepping, full electrostatic evaluation independent of cycle length, and the ability to do on-the-fly analysis and apply forces to the system via Tcl scripts. The new multiple timestepping parameter `nonbondedFreq` allows nonbonded interactions to be evaluated less frequently, leaving only bonded interactions to be calculated every timestep. The new spatial partitioning parameters `splitPatch` and `hgroupCutoff` have been added due to a change from pairlists to group-based distance checking. The Langevin dynamics and temperature coupling algorithms has been altered to be more suitable for temperature control with small damping coefficients. Finally, NAMD 2.0 can now read CHARMM as well as X-PLOR formatted parameter files.

Several options have been removed in this release. Multiple timestepping is currently limited to impulse methods. Margin checking is performed at every timestep. There are not separate switching and cutoff parameters for van der Waals and electrostatic potentials. Less-used features such as an applied electric field and force DCD files are unimplemented.

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, 7]. In order to conduct MD simulations, various computer programs have been developed including X-PLOR [5] and CHARMM [4]. 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 [4] and X-PLOR [5]. 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 Distributed Parallel Multipole Tree Algorithm (DPMTA) [3] and Distributed Particle Mesh Ewald (DPME) algorithms, which takes the full electrostatic interactions into account. These algorithm reduces the computational complexity of electrostatic force evaluation from $O(N^2)$ to $O(N)$ or $O(N \log N)$.

- **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 [2], 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. The structure and design of the program is described in the NAMD *Programmer's Guide* with sufficient detail to allow additions of such new algorithms.
- **Interactive MD simulations**
  NAMD is one component of MDScope, a computational environment for structural biology. The two other components of MDScope are VMD, a molecular visualization program, and MDComm, a communications package. When combined, 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 MDScope, see `http://www.ks.uiuc.edu/Research/mdscope/`.
- **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 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 or the NAMD *Programmer's Guide*, 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. Please take a few minutes to complete the short survey on the next page and mail or fax it to

NAMD Support
Theoretical Biophysics Group
Beckman Institute
University of Illinois
405 N. Mathews St.
Urbana, IL 61801
FAX (217) 244-6078

or email the relevant information to `namd@ks.uiuc.edu` with subject "Survey." This will help us make NAMD relevant and responsive to the needs of our user community.

1. Name:
2. Company/University:
3. Address:
4. Email address:
5. Research area:
6. Molecular systems to be studied with NAMD:
7. NAMD features most useful for your research:
8. I run NAMD on the following platforms and operating systems:
9. I would like to run NAMD on:
10. How can we improve the NAMD documentation?
11. Other comments:

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 8, 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.

We strongly recommend that you have access to either CHARMM or X-PLOR, either of which is capable of generating both the PSF and PDB files. NAMD currently provides no automatic method of generating these 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 2.2.2 lists the parameters which are required to run a basic simulation. Section 6 describes the relation between specific NAMD and X-PLOR dynamics options. Several sample NAMD configuration files are shown in section 7.

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 Required NAMD configuration parameters

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

- `numsteps` (page 23),
- `coordinates` (page 15),
- `structure` (page 15),
- `parameters` (page 15),
- `exclude` (page 25),
- `outputname` (page 16),
- one of the following three:
  - `temperature` (page 25),
  - `velocities` (page 16),
  - `binvelocities` (page 16).

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 [5] and CHARMM [4]. 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 [5].

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.

- `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 be included in the trajectory file unless the `firsttimestep` parameter is set non-zero.

- `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 be included in the trajectory file unless the `firsttimestep` parameter is set non-zero.

- `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**.

- `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.

4 Basic Simulation Parameters

4.1 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.

4.1.1 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.

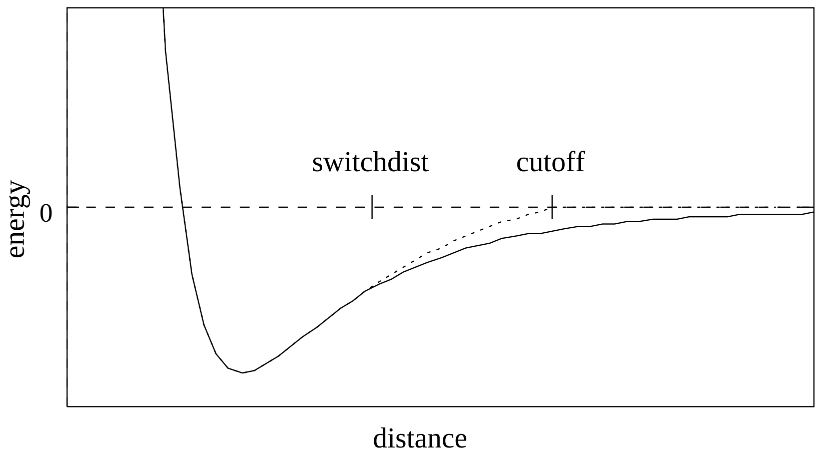

Figure 1: Graph of van der Waals potential with and without the application of the switching function. With the switching function active, the potential is smoothly reduced to 0 at the cutoff distance. Without the switching function, there is a discontinuity where the potential is truncated.

The details of the switching function are given in the NAMD *Programmer's Guide*. 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`.

4.1.2 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 fully described in the NAMD *Programmer's Guide* and is based on the shifting function used by X-PLOR.

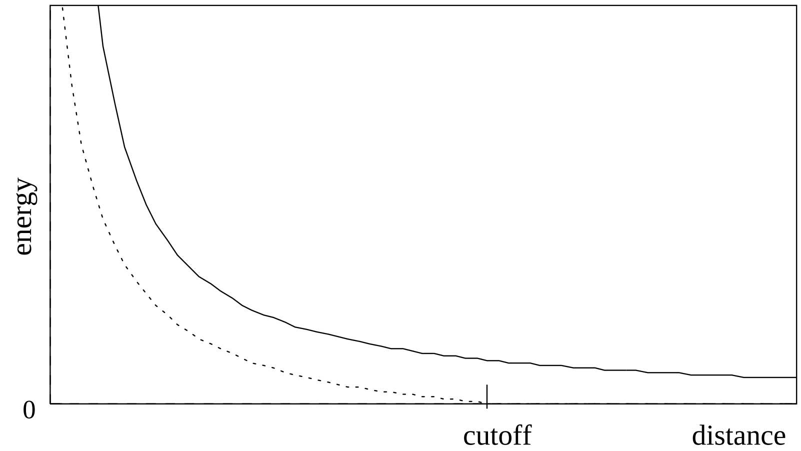

Figure 2: Graph showing an electrostatic potential with and without the application of the shifting function.

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 4.2.

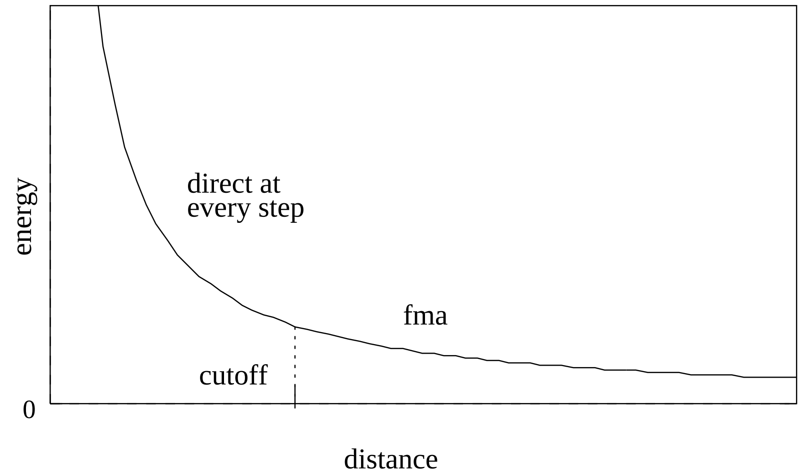

Figure 3: Graph showing an electrostatic potential when full electrostatics are used within NAMD, with one curve portion calculated directly and the other calculated using DPMTA.

4.1.3 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.

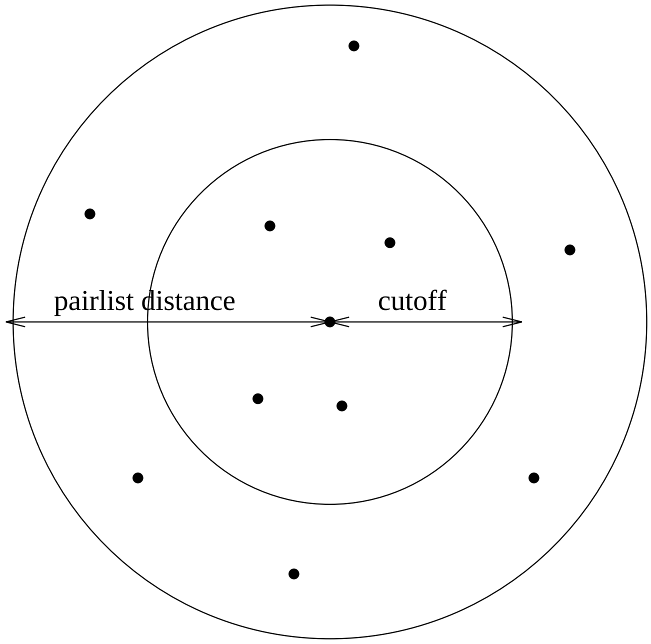

Figure 4: Depiction of the difference between the cutoff distance and the pair list distance. The pair list distance specifies a sphere that is slightly larger than that of the cutoff so that pairs are allowed to move in and out of the cutoff distance without causing energy conservation to be disturbed.

4.2 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, and each set of k timesteps is referred to as a *cycle*. The value of k is specified by the NAMD parameter `stepspercycle`. 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.

4.3 NAMD configuration parameters

4.3.1 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` × `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 pairlist generation and atom reassignment. If full electrostatics are active, it is also the number of timesteps between full electrostatic evaluation unless `fullElectFrequency` is also specified. It is recommended that `stepspercycle` (`fullElectFrequency`) be chosen so that the product of `stepspercycle` and `timestep` does not exceed 4.0 unless `rigidBonds all` is specified, in which case the upper limit is perhaps doubled. For more details on the use of full electrostatics, see Section 4.2. For more details on non-bonded force evaluation and pairlist generation, see Section 4.1.

4.3.2 Simulation space partitioning

- `cutoff` < local interaction distance common to both electrostatic and van der Waals calculations (Å) >
  Acceptable Values: positive decimal
  Description: See Section 4.1 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 4.1. 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 ≤ `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 4.1.

- `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: 1.0
  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. For more details about this parameter see the NAMD *Programmer's Guide*. Unless you are very motivated to get the *very* best possible performance, just leave this value at the default.

4.3.3 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.

4.3.4 DPMTA parameters

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 4.2. 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 4.2.

- `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.

4.3.5 DPME parameters

DPME stands for Distributed 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 DPME.

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

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

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

4.3.6 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 DPME 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.

4.3.7 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. The parameters in this Section are part of this exploration. These parameters are still rather experimental and are not fully documented yet, but the brave at heart may try them. By default, the naive scheme described in the NAMD *Programmer's Guide* is used.

- `fullElectFrequency` < number of timesteps between DPMTA calculations >
  Acceptable Values: positive integer factor of `stepspercycle`

Default Value: `stepspercycle`
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` 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: `verletI`
Default Value: `verletI`
Description: Specifies the multiple timestep algorithm used to integrate the long and short range forces. `verletI` is the same as r-RESPA.

- `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 [6]:

$$SW(r_{ij}) = 0 \text{ if } |\vec{r}_{ij}| > R_{\mathit{off}}$$
$$SW(r_{ij}) = 1 \text{ if } |\vec{r}_{ij}| \leq R_{on}$$
$$SW(r_{ij}) = \frac{\left({R_{\mathit{off}}}^2 - {r_{ij}}^2\right)^2 \left({R_{\mathit{off}}}^2 + 2{r_{ij}}^2 - 3{R_{on}}^2\right)}{\left({R_{\mathit{off}}}^2 - {R_{on}}^2\right)^3} \text{ if } R_{\mathit{off}} > |\vec{r}_{ij}| \geq R_{on}$$

where

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

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

5 Additional Simulation Parameters

5.1 Constraints and Restraints

5.1.1 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. For further details, see the NAMD *Programmer's Guide*.

- `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.

5.1.2 Fixed atoms parameters

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

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

- `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: `O`
  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.

5.2 Energy Minimization

5.2.1 Velocity quenching parameters

As described in the NAMD *Programmer's Guide*, NAMD has the capability of performing 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.

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

5.3 Temperature Control and Equilibration

5.3.1 Langevin dynamics parameters

As described in the NAMD *Programmer's Guide*, 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* [5], 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.

5.3.2 Temperature coupling parameters

As described in the NAMD *Programmer's Guide*, 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* [5].

- `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.

5.3.3 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.

5.3.4 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.

5.4 Boundary Conditions

5.4.1 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 as described in the NAMD *Programmer's Guide*. 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 `sphericalBCr1` and `sphericalBCk1` must be defined, and `sphericalBCexp1`, `sphericalBCr2`, `sphericalBCk2`, and `sphericalBCexp2` can optionally be defined.

- `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 of mass.

- `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 of mass, and a negative value will pull atoms away from the center of mass.

- `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 of mass. 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 of mass, and a negative value will pull atoms away from the center of mass.

- `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.

5.4.2 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 as described in the NAMD *Programmer's Guide*. 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 `cylindricalBCr1`, `cylindricalBCl1` and `cylindricalBCk1` must be defined, and `cylindricalBCAxis`, `cylindricalBCCenter`, `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. If not specified, center of mass is used.

- `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.

5.4.3 Periodic boundary conditions

NAMD provides periodic boundary conditions in 1, 2 or 3 dimensions. The basis vectors of the periodic cell must be along the x, y, and z axes. This restriction will be eliminated in future versions of NAMD. The following parameters are used to define these boundary conditions.

- `cellBasisVector1` < basis vector for periodic boundaries (Å) >
  Acceptable Values: vector along x axis
  Default Value: 0 0 0
  Description: Specifies whether or not periodic boundary conditions are to be applied to the system in the x direction.

- `cellBasisVector2` < basis vector for periodic boundaries (Å) >
  Acceptable Values: vector along y axis
  Default Value: 0 0 0
  Description: Specifies whether or not periodic boundary conditions are to be applied to the system in the y direction.

- `cellBasisVector3` < basis vector for periodic boundaries (Å) >
  Acceptable Values: vector along z axis
  Default Value: 0 0 0
  Description: Specifies whether or not periodic boundary conditions are to be applied to the system in the z direction.

- `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. This option alters this behavior for water molecules only.

5.5 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. This is not currently implemented in Berendsen's method.

5.5.1 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.

5.5.2 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.

- `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`.

5.6 Applied Forces and Analysis

Currently, there are two ways to steer simulations with NAMD. One is called "moving constraints" and is based on the harmonic constraints feature. The other is a stand-alone feature called "SMD". In both cases the user specifies a reference position $\vec{r}_0$, force constant k, velocity of the reference position movement $\vec{v}$ and the number of the constrained atom i. Then during the simulation a potential U_{SMD} and a corresponding force $\vec{f}_{SMD}$ are added for the specified atom. The potential is computed according to

$$U_{SMD}(t) \;=\; \frac{k\,[\vec{r}(t) \;-\; \vec{r}_i(t)]^2}{2}\,, \tag{1}$$

and the force acting on atom i is (by differentiating U_{SMD})

$$\vec{f}_{SMD}(t) \;=\; k\,[\vec{r}(t) \;-\; \vec{r}_i(t)]\,, \tag{2}$$

where t is time, $\vec{r}_i$ is the position of the atom i and $\vec{r}$ is the current reference position (position of the restraint point), given by

$$\vec{r}(t) \;=\; \vec{r}_0 \,+\, \vec{v}t\,. \tag{3}$$

5.6.1 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 Eq. 3. 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. 3, 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. This feature allows the restraint reference positions for *all restrained atoms* to be moved. Moving the restraint reference positions for a single atom is provided by the SMD feature.

- `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

5.6.2 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 $= \Omega t$, 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 $= F \times normal$ Finally, the torque applied to the atom with respect to the axis is the projection of the torque on the axis, i.e., $torque_{proj} = torque \cdot v$

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.

5.6.3 Steered Molecular Dynamics (SMD)

The SMD feature is independent from the harmonic constraints, although it follows the same ideas. One has to specify the force constant k (`SMDk`), the number (1-based indexing) of the atom (`SMDAtom`), which is restrained to the moving reference position, the initial reference position $\vec{r}_0$ (`SMDRefPos`), the *absolute* value of the velocity of movement v (`SMDVel`), and the direction of movement $\vec{n}$ (`SMDDir`). The velocity $\vec{v}$ is then given by $\vec{v} = v\vec{n}$. Vector $\vec{n}$ is normalized by NAMD before being used. Optionally, the frequency of SMD data output can be specified.

The time used in the reference position calculation starts at time t_0 which can be specified through `SMDTStamp` parameter, defaulting to `firstTimestep`. The reference position is calculated then by

$$\vec{r} = \vec{r}_0 + \vec{n}v\,(t - t_0)\,, \tag{4}$$

where t is the current timestep (i.e., `firstTimestep` plus however many timesteps have passed since the beginning of NAMD run). When restarting the simulation it may be useful to set t_0 to whatever time the reference position specified in the configuration file refers to.

Changing direction of reference position movement One may want to change the direction of the reference position movement, e.g., if the restrained atom is moving too slow in the given direction. To check for such slow movement, the minimum allowed average velocity v_{min} (`SMDVmin`) and the averaging time τ_{min} (`SMDVminTave`) have to be specified. Then every τ_{min} timesteps NAMD will compute the average velocity in the current direction over the past τ_{min} timesteps, and compare it to v_{min}. The average velocity is computed simply by

$$\langle v \rangle_{min} \;=\; \frac{\vec{n} \cdot [\vec{r}_i(t) \;-\; \vec{r}_i(t - \tau_{min})]}{\tau_{min}} \,. \tag{5}$$

In the case that $\langle v \rangle_{min}$ is less than v_{min}, a new direction is chosen, and the simulation proceeds.

Every time the direction is changed, the reference point is also changed to

$$\vec{r}_{0,new} \;=\; \vec{r}_i(t) + \vec{n}_{new} F_{min}/k \,, \tag{6}$$

where t is the current timestep, and F_{min} (`SMDFmin`) is the minimum force to which the force imposed by the restraint is reset. Thus, after the direction is changed to $\vec{n}_{new}$ the force applied to the atom has an absolute value of F_{min} and the direction of $\vec{n}_{new}$. The time stamp t_0 is set to current time.

Choice of new direction The choice of the new (random) direction is based on the specified initial direction $\vec{n}$, the angle θ_c of the cone which has $\vec{n}$ as its axis, and the method by which the random numbers are generated. The new direction is guaranteed to lie within the specified cone. The method can be "uniform" (default) or "gaussian". Uniform method uses uniformly distributed random numbers to find angles φ and θ defining the new direction. Angle φ is a random number between 0 and 360 degrees, and θ is a random number between 0 and $\theta_c/2$. Gaussian method uses uniformly distributed random numbers to find φ and normally distributed random numbers with variance σ (`SMDGaussW`) to find θ. All θ values that exceed $\theta_c/2$ are ignored. The distribution of θ values is given by

$$p(\theta) \;=\; \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{\theta^2}{\sigma^2}) \,. \tag{7}$$

After the angles φ and θ are calculated, the new direction is found by constructing a right orthonormal system of coordinates with $\vec{n}_0$, the initial direction specified by `SMDDir` in the configuration file, being the z direction. The other two vectors $\vec{a}$ and $\vec{b}$, orthogonal to $\vec{n}_0$ and to each other are found by first producing $\vec{a} \;=\; \vec{n}_0 \times (0, \vec{1}, 1)$, or $\vec{a} \;=\; \vec{n}_0 \times (1, \vec{1}, 1)$ in the case when $\vec{n}_0$ is collinear to $(0, \vec{1}, 1)$, normalizing $\vec{a}$, and, finally, producing $\vec{b} \;=\; \vec{n}_0 \times \vec{a}$. Then the new direction is given by

$$\vec{n}_{new} \;=\; \vec{a} \cos\varphi \sin\theta \;+\; \vec{b} \sin\varphi \sin\theta \;+\; \vec{n}_0 \cos\theta \,. \tag{8}$$

Resetting the applied force Sometimes, especially when a small enough k is employed, the movement of the restrained atom proceeds in steps, resulting in a fast movement of the atom over a short period of time. When this happens, it may be desirable to reset (reduce) the applied force. Resetting of the force to a specified value F_{min} (`SMDFmin`) should happen when the average velocity of atom movement $\langle v \rangle_{max}$ over a given period of time τ_{max} (`SMDVmaxTave`) exceeds the specified maximum allowed velocity v_{max} (`SMDVmax`) in the direction of reference position movement $\vec{n}$. The average velocity is calculated similarly to Eq. 5, and in the case when $\langle v \rangle_{max}$ is larger than v_{max}, the reference position is reset according to Eq. 6. The time stamp t_0 is set to current time.

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 restrained atom, the current force applied to the restrained atom (in piconewtons, pN), the reference position at the time of the last change of direction or force resetting (initially, the reference position specified in the configuration file), current direction of the reference position movement, and the time stamp, indicating the timestep when the direction was changed or force reset (initially, `SMDTStep` specified in the configuration file, or its default value). The output line starts with word `SMD`

The energy of the SMD restraint is printed out along with other energies in the column titled `SMD`. The frequency of this output is defined by `outputEnergies` parameter in the configuration file.

When the direction of the reference position movement changes or the force is reset, this change is indicated by an output line starting with `SMDChange` and showing the current timestep, the current atom position, the new reference position after the change, and the new direction of the reference position movement. The current timestep becomes the time stamp for the following simulation until the next change occurs.

Parameters The following parameters describe the parameters for the SMD feature of NAMD. This feature allows a harmonic restraining force to be applied to any ONE atom in the simulation and the restraining reference position to be moved. It also allows to randomly change the direction of the reference position movement if the restrained atom progress is too slow, and to reset the applied force to a given value if the progress of the restrained atom is too fast.

- `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`, `SMDRefPos`, `SMDVel`, `SMDDir` and `SMDAtom` must be defined. Also, `SMDOutputFreq`, `SMDChDir` and associated parameters, `SMDChForce` and associated parameters, and `SMDTStamp` can be optionally defined.

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

- `SMDRefPos` < SMD constraint reference position >
  Acceptable Values: vector
  Description: Vector to use for the initial reference position for the SMD harmonic constraints. The atom that is specified by `SMDAtom` will be initially constrained to this reference position. During the simulation, this reference position 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: positive real, Å/timestep

Description: The velocity of the SMD reference position movement. Gives the absolute value.

- `SMDDir` < Direction of the SMD reference position 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.

- `SMDAtom` < Index (1-based) of the atom constrained to the moving reference position >
  Acceptable Values: positive integer
  Description: The index of the atom constrained to the moving reference position. If the atom is numbered N in the PDB file, N must be specified.

- `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.

- `SMDTStamp` < time of the last change in reference position >
  Acceptable Values: non-negative integer
  Default Value: `firstTimestep`
  Description: The timestep at which the reference position $\vec{r}_0$ was last reset. Normally, it would be the `firstTimestep`, but if the force or direction was reset during the simulation, this parameter may be useful for restarts.

- `SMDFmin` < value of the force to reset to >
  Acceptable Values: non-negative real
  Default Value: 0
  Description: The value in pN to which the force gets reset when the direction changes or the average velocity exceeds the given `SMDVmax` value. This parameter is only used if `SMDChDir` or `SMDChForce` are `on`.

- `SMDChDir` < Is changing direction option of SMD active >
  Acceptable Values: `on`, `off`
  Default Value: `off`
  Description: If turned `on`, this option allows to change the direction of the reference position movement when the average velocity of the restrained atom in this direction is smaller than the given `SMDVmin` value. Uses settings of `SMDVmin`, `SMDVminTave`, `SMDConeAngle`, `SMDChDirMethod`.

- `SMDVmin` < minimum allowed average velocity of the restrained atom >
  Acceptable Values: non-negative real
  Description: The minimum allowed average velocity in Å/timestep of the restrained atom in the direction of the reference position movement. The averaging time is given by `SMDVminTave`.

- `SMDVminTave` < averaging time for velocity to compare to Vmin >
  Acceptable Values: positive integer

Description: The averaging time in timesteps for calculation of the average velocity of the restrained atom and for comparison of this average velocity to `SMDVmin`.

- `SMDConeAngle` < angle of the cone to choose the new direction from >
  Acceptable Values: non-negative
  Default Value: 360
  Description: The angle (in degrees) of the cone with the axis along the initial direction of the reference position movement `SMDDir`. The new direction will come from the vertex of the cone and will lie within the cone. This angle is twice the angle between the cone's axis and any line going through the cone's vertex on the cone's surface. Omitting this parameter will lead to choosing a direction without any restrictions (all directions are possible).

- `SMDChDirMethod` < method to use when choosing a new direction >
  Acceptable Values: `gaussian`, `uniform`
  Default Value: `uniform`
  Description: The method to choose the angle between the new direction and the initial direction of the reference position movement. When `gaussian` is specified, it is recommended to set the variance of the distribution with `SMDGaussW`.

- `SMDGaussW` < variance of the gaussian distribution used to choose new directions >
  Acceptable Values: positive real
  Default Value: 360
  Description: Variance of the gaussian distribution (in degrees) used for choosing the angle between the new direction and the initial direction of the reference position movement.

- `SMDChForce` < is resetting force SMD option active >
  Acceptable Values: `on`, `off`
  Default Value: `off`
  Description: If turned `on`, this option allows to reset the reference position (so that the applied force becomes `SMDFmin`) when the average velocity of the restrained atom in the direction of the reference position movement is larger than a given `SMDVmax` value. Uses settings of `SMDVmax` and `SMDVmaxTave`.

- `SMDVmax` < maximum allowed average velocity of the restrained atom >
  Acceptable Values: non-negative real
  Description: The maximum allowed average velocity in Å/timestep of the restrained atom in the direction of the reference position movement. The averaging time is given by `SMDVmaxTave`.

- `SMDVmaxTave` < averaging time for velocity to compare to Vmax >
  Acceptable Values: positive integer
  Description: The averaging time in timesteps for calculation of the average velocity of the restrained atom and for comparison of this average velocity to `SMDVmax`.

5.6.4 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 <molname> <resid> <atomname>`
  Determines atomid of an atom from its molecule, 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 `reconfig` 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 `reconfig` is called. For example, "`set groupid [addgroup  14 10 12 ]`".

- `reconfig`
  Signals that new atoms are being requested. `addatom` and `addgroup` calls during `calcforces` will be ignored unless `reconfig` is called. Old configuration is replaced by new configuration. `reconfig` should only be called from within the `calcforces` procedure.

- `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)`".

- `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.

5.7 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.

5.7.1 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 $\overrightarrow{r_{ref}}$
$$E = (K_f/2)\,(|\overrightarrow{r_i} - \overrightarrow{r_{ref}}|)^2$$
Stretch restraint (2 atoms): force constant K_f, and reference distance d_{ref}
$$E = (K_f/2)\,(d_i - d_{ref})^2$$
Bend restraint (3 atoms): force constant K_f, and reference angle θ_{ref}
$$E = (K_f/2)\,(\theta_i - \theta_{ref})^2$$
Torsion restraint (4 atoms): energy barrier E_0, and reference angle χ_{ref}
$$E = (E_0/2)\,\{1 - \cos(\chi_i - \chi_{ref})\}$$
Forcing restraints
Position restraint (1 atom): force constant K_f, and two reference positions $\overrightarrow{r_0}$ and $\overrightarrow{r_1}$
$$E = (K_f/2)\,(|\overrightarrow{r_i} - \overrightarrow{r_{ref}}|)^2$$
$$\overrightarrow{r_{ref}} = \lambda\overrightarrow{r_1} + (1-\lambda)\,\overrightarrow{r_0}$$
Stretch restraint (2 atoms): force constant K_f, and two reference distances d_0 and d_1

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

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

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

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

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

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, $\Delta\chi$.

$$\begin{array}{llll}
\chi_1 & < \chi & < \chi_2 : & E = 0 \\
(\chi_1 - \Delta\chi) & < \chi & < \chi_1 : & E = (G/2)\,\{1 - \cos(\chi - \chi_1)\} \\
\chi_2 & < \chi & (\chi_2 + \Delta\chi): & E = (G/2)\,\{1 - \cos(\chi - \chi_2)\} \\
(\chi_2 + \Delta\chi) & < \chi & (\chi_1 - \Delta\chi + 2\pi) : & E = G
\end{array}$$
$$G = E_0 / \{1 - \cos(\Delta\chi)\}$$

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

5.7.2 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.

5.7.3 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)

5.7.4 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) }

5.7.5 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 $\rightarrow$1 or 1 $\rightarrow$0, or b) set lambda. The mcti block is used to vary λ from 0 $\rightarrow$1 or 1 $\rightarrow$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 λ_t; 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 $\rightarrow$1. (no value of λ is required)
down, fade:	λ changes from 1 $\rightarrow$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 λ_t; 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 $\rightarrow$1. (no value of λ is required)
stepdown, stepfade:	λ changes from 1 $\rightarrow$0. (no value of λ is required)

For each task, λ changes in steps of (1.0/N) from 0 $\rightarrow$1 or 1 $\rightarrow$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.

5.7.6 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^o .
  // 4. restrain the dihedral angle between the ca's of residues 0-10-20-30 to 180^o .
  // 5. restrain the angle between the centers-of-mass of residues 0-10-20 to 90^o .
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^o
  //      ( (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^o  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^o   ¡ χ   ¡ 120^o
  // The energy is 20 kcal/mol for:       130^o   ¡ χ   ¡ 260^o
  // Energy rises from 0 → 20 kcal/mol as χ increases from 120^o → 130^o , and decreases from –90^o → –100^o.
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
}
```

5.7.7 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

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

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

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

for atom 2 moving, and atom 1 fixed

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

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

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

Gradient for bend restraint

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

Atoms at positions A-B-C

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Gradient for dihedral angle restraint

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

Atoms at positions A-B-C-D

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

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

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

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

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

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

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

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

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

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

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

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

Gradient for forcing position restraint

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

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

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

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 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.

7 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.

```
timestep        0.5
numsteps        10000

cwd             /scratch                #  Specify a working directory

structure       alanin.psf
parameters      alanin.params
coordinates     alanin.pdb

exclude         scaled1-4
1-4scaling      0.4
outputname      output
margin          1.0
stepspercycle   10
temperature     300.0

switching       on
switchdist      7.0
cutoff          8.0
pairlistdist    9.0
```

This file is again for alanin, but shows a slightly more complicated configuration. A coordinate trajectory file and a set of restart files are produced every ten timesteps. Langevin dynamics are also active in this configuration.

```
timestep        0.5

numsteps        10000

structure       alanin.psf
parameters      alanin.params
coordinates     alanin.pdb
exclude         scaled1-4
1-4scaling      0.4
outputname      output
margin          1.0
stepspercycle   10
temperature     300.0

switching       on
switchdist      7.0
cutoff          8.0
pairlistdist    9.0

DCDfile         alanin.dcd
DCDfreq         10

restartname     alanin.restart
restartfreq     10

langevin        on
langevinTemp    300.0
langevinCol     O
```

This file shows another simple configuration file for alanin, but this time with full electrostatics using DPMTA.

```
timestep        0.5
numsteps        10000

cwd             /scratch        #  Specify a working directory

structure       alanin.psf
parameters      alanin.params
coordinates     alanin.pdb

exclude         scaled1-4
1-4scaling      0.4
outputname      output
margin          1.0
stepspercycle   10
temperature     300.0

switching       on
switchdist      7.0
cutoff          8.0
pairlistdist    9.0

# DPMTA parameters
FMA             on
FMAMp           8
FMALevels       3
FMAFFT          on
FMAFFTBlock     4
```

8 Running NAMD

NAMD currently runs on network of HP, Sun, SGI, and Linux workstations, as well as Cray T3E, IBM SP3, and SGI-CRAY Origin 2000 machines. On network of workstations, a "host program" (called `conv-host`) is needed to launch NAMD on individual workstations in the network. This host program is included with the distribution. The general command line syntax for invoking NAMD on each of these platforms is:
`<hostprog> namd2 <config-file> +p`N
Where `<config-file>` is the name of the configuration file. And N is the number of processors to be used for execution.

8.1 Platform-Specific Notes

The following subsections explain in detail the steps involved in running NAMD on particular platforms.

8.1.1 Network of Workstations

In order to run NAMD on network of workstations, a host program (called `conv-host`) is needed in order to launch NAMD on individual workstations. This program is bundled with the binary distribution. The names of individual workstations as well as other control information such as how many processes to create on those workstations (should they happen to be multiprocessor workstations) is specified to the host program through a file named `.nodelist` in the user's home directory or can be superseded by the `nodelist` file in the current directory. Details about the syntax of `nodelist` file can be found in *Converse Installation and Usage Manual* at `http://charm.cs.uiuc.edu/`.

8.1.2 IBM SP3

On IBM SP3, you would have to use the scheduler available on the front end of the SP system in order to run NAMD. Below, we give the description of the commands we use at Argonne National Laboratory's IBM SP3 installation. Consult the system documentation or your local system administrator about the details at your site. We use the command `spsubmit` to submit NAMD runs to the SP scheduler at Argonne National Laboratory.
`spsubmit -np` N `-maxtime` *mins* `namd2 <config-file>`
N is the number of processors we request NAMD to run on, and *mins* i the maximum time we expect NAMD to take during this run.

8.1.3 CRAY T3E

`mpprun` is the name of the command used to run parallel jobs on CRAY T3E. This command can be used as:
`mpprun namd2 <config-file> +p`N
where N is the number of requested processors. However, this can be used only when the number of requested processors is less than or equal to 32. For jobs larger than this, one has to use queuing facility provided at the site of installation. Description of the queuing facilities is out of this document's scope, and we leave that up to you to explore.

8.1.4 Origin2000

Origin2000 version of NAMD is a shared memory version, and does not need any particular *host* program to launch NAMD. Thus, in order to run NAMD on Origin2000, we use the following command:
`namd2 <config-file> +p`N
where, N is the number of processes we plan to use. However, note that depending on the load and the queuing strategy employed by the system some of the processes may be interleaved on some processors, thus causing the performance to suffer. Some installations of Origin2000 employ an external queuing facility to avoid this problem. You are strongly encouraged to consult your local documentation for queuing facility used.

8.2 Interactive modeling with MDScope

Interactive molecular modeling can be performed using the MDScope environment. MDScope consists of NAMD; VMD, a program for interactive visualization of biopolymers; and MDComm, a communications library for efficient network transfer of molecular dynamics information. VMD allows the user to view intermediate results of the simulation being performed by NAMD while the simulation is still proceeding. For more information regarding the installation or use of MDScope, consult the VMD documentation.

9 NAMD Availability and Installation

NAMD is distributed freely for non-profit use. NAMD 2.0 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 2.0.

9.1 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. Pre-compiled binaries are available for most architectures to which NAMD has been ported.

9.2 Platforms on which NAMD will currently run

NAMD 2.0 should be portable to any parallel platform with a modern C++ compiler to which Charm and Converse have been ported. Some minor tuning of system parameters or compiler flags may be required. NAMD 2.0 has been compiled and tested on the following platforms:

- Intel PC's (Linux)
- HP PA-RISC workstations
- Sun UltraSparc workstations
- ACSI Red
- IBM SP3
- Cray T3D and T3E
- SGI Origin 2000, Origin 200, and Onyx 2

9.3 Compiling NAMD

To compile NAMD for a particular machine, follow these steps:

1. Obtain the latest release of Charm from `http://charm.cs.uiuc.edu/` and compile it for the platforms you wish to use.
2. Edit `Make.charm` so that the `CHARMBASE` variable points to the top directory of your Charm installation.
3. The `arch` directory contains files of the form `Makearch.<architecture>`. Find the file which is closest to the platform you wish to compile for and edit it. You may need to alter some of the following:
    - `CXX` - C++ compiler
    - `CXXOPTS` - C++ compiler options
    - `CC` - C compiler
    - `COPTS` - C compiler options

- `CHARMARCH` - Charm architecture name
- `TCL...` - definitions for Tcl installation (optional)
- `MDCOMM...` - definitions for MDComm installation (very optional)

4. In the main directory run `../config <architecture>` which will create a new build subdirectory `<architecture>`. You only need to repeat this step if you add new source files.

5. Change to this new directory and type `make`.

9.4 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] 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.

[3] J. Board, Z. Hakura, W. Elliot, and W. Rankin. Scalable variants of multipole-accelerated algorithms for molecular dynamics applications. Technical Report TR94-006, Duke University, Dept. of Elec. Engr., 1994.

[4] 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.

[5] 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.

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

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