PROTOMOL Version 2.0.3 - Quick Reference

June 24, 2005

Chapter 1

Introduction

Molecular Dynamics (MD) describes a molecular system as a function of time based on integration of equations of motion. The most computationally expensive part is the force calculation at every time step. There have been many implementations to solve one given problem very efficiently, but they usually lack flexibility when trying to implement new methods or approaches to solve the problem. What PROTOMOL provides is a generic, object-oriented component framework for MD simulations. To meet these high performance expectations, PROTOMOL uses cell algorithms, grid techniques and well-optimized libraries for the most computationally expensive forces. The design of PROTOMOL also includes parallelization, based on components to distribute the work and data. The approach follows an incremental and partial parallelization scheme, which allows the developer to start with a sequential implementation and then do step by step parallelization.

The overall framework of PROTOMOL is designed for non-bonded, bonded, short-range and long-range forces for systems with tens of thousands of atoms representing water and several large molecules. It is designed for high flexibility, ease in extension and maintainence, and high performance demands.

1.1 Useful Features

1.1.1 Force field

PROTOMOL supports force fields in the same format that CHARMM uses, including bonded interactions between groups of 2 (bonds), 3 (angles), and 4 (dihedrals and impropers) atoms, as well as electrostatic and van der Waals nonbonded interactions.

1.1.2 Multiple time-stepping

Multiple time-stepping is a useful technique for cost reduction of calculating long-range electrostatic forces. The idea behind multiple time-stepping is to compute at every timestep bonded, van der Waals and short-range electrostatic forces, while computing long-range electrostatic forces less often. Overtime this will result in improved efficiency because the cost for computing electrostatic interactions will be amortized over the number of timesteps to be run in the simulation [1].

1.1.3 Comparison of force algorithms

PROTOMOL is able to compare forces measuring the error and the time; e.g. an exact algorithm with a fast approximation.

1.1.4 Ability to interact with VMD

VMD is a program developed by the University of Illinois that is used for displaying large biomolecular systems in three dimensions. PROTOMOL is conveniently able to interact with this useful program. For more information, see:

`http://www.ks.uiuc.edu/Research/vmd`

1.1.5 Ability to run on parallel machines

The most computationally expensive part of an MD simulation is force evaluation between atoms. Because so much computing power is required for the force evaluation of large systems, *i.e.*, 5000 atoms or more, the ability to run MD simulations on parallel machines will hold a large advantage with regard to speed. PROTOMOL has been designed to take advantage of parallel computing in running MD simulations.

1.1.6 Installation

Please follow the README file under protomol directory for installation instructions.

1.1.7 Licensing

PROTOMOL is a free software, available under GNU general public license(GPL) [1]. Please cite [10] if you are using PROTOMOL for your research.

[1] http://www.gnu.org/licenses/licenses.html#GPL

Chapter 2

Getting Started

This chapter will introduce the commands and settings needed to run PROTOMOL. Included are the exact formats of the command line on a UNIX / Linux machine and the configuration file containing all initial information to run the MD simulation. In Chapter 4 we will show three sample configuration files.

2.1 Command Line

The MD application of the PROTOMOL framework has conveniently been named `protomol`. At a UNIX / Linux prompt, a user types `protomol` followed by an alternating list of keywords and arguments (see chapter 3 for a list of keywords). Thus, the general format for the PROTOMOL execution command is the following:

`protomol [--keyword1` *value1* `] [--keyword2` *value2* `] [--keyword3` *value3* `] .......`

Note that keywords must be preceded by two dashes, where a value can be a list of values - e.g. for the keyword CellBasisVector1. Also note that any keyword-value pair specified on the command line overrides any according pair in the configuration file.

The user may also specify any of these keywords and values in the configuration file that he or she is using. The exact pathname of the configuration file being used for running PROTOMOL must be specified on the command line. For example, the configuration file can be specified one of the following ways:
`protomol [--config] <pathname/myconfigfilename> ......`

The `--config` *keyword can be omitted if the configuration file is the first thing specified.*

```
protomol .....  --config <pathname/myconfigfilename> .....
```

2.1.1 Help Option

A user may type one of the following two possibilities at a UNIX / Linux prompt for help with the PROTOMOL command line:

1. `protomol -h`
2. `protomol --help`

2.2 Simple GUI

In oder simplify the usage of PROTOMOL (i.e., on Windows) you may use the simple perl/Tk GUI front-end `TkProtomol.pl`. The script is located under `protomol/tools`, and has a simple editor and plotter.

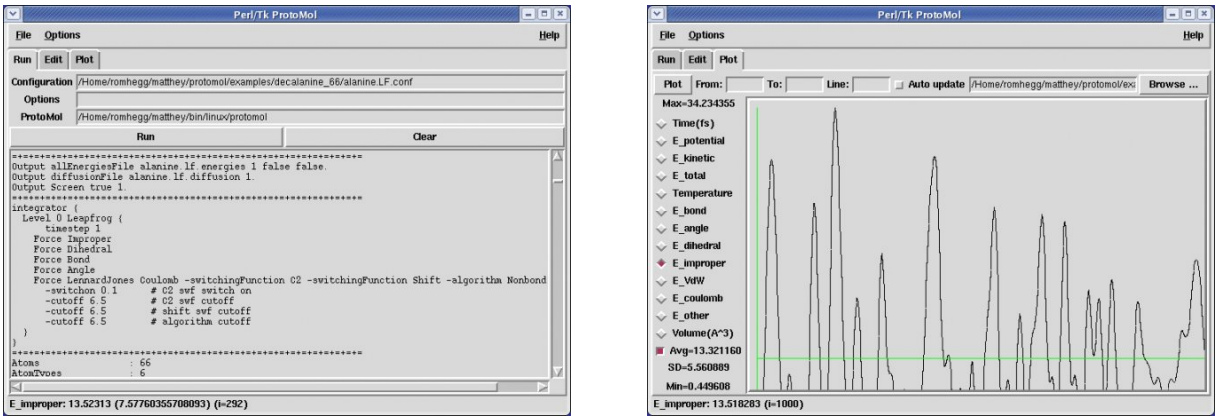

Figure 2.1: Simple perl/Tk GUI front-end.

2.3 Configuration File

The configuration file is a text file containing a collection of keyword-value pairs specifying the simulation configuration, I/O files and formats, and the definition of the integrator scheme.

2.3.1 Format of PROTOMOL keywords in the configuration file

The configuration file format for PROTOMOL is quite simple, making it convenient for creation and modification of the file. The advantage of straightforward modification of the configuration file is that the user can very easily switch between running the same molecule under different initial conditions, or even switch to a different molecule without much trouble. The general format for a PROTOMOL configuration file is a list of keywords and values, with whitespace between each keyword and value and a newline between each new keyword-value pair:

keyword1 *value1*
keyword2 *value2* # comment
comment
keyword3 *value3*
.
.
.

A list of PROTOMOL *keywords, possible values and defaults can be found in section 3, or type* `protomol -m --keywords`.

2.3.2 Format of the integrator and their arguments

In addition to the keyword-value pairs in the configuration file, one must set up an integrator in the following manner:

Integrator {
  level N-1 <integrator type>{ # MTS integrator
    <integrator arguments> *(These will differ depending on the integrator type).*
    <integrator forces> *(These are all optional).*
  }
  .
  .
  .

  level 0 <integrator type>{ # STS integrator
    .
    .
    .
  }
}

Note that the order of definition for each level is not strict, but PROTOMOL expects one definition for each level.

Integrator Types

- Multiple Timestep Integrators (MTS)
  1. **BSplineMOLLY**
  2. **EquilibriumMOLLY**
  3. **HybridMC** *(Hybrid Monte Carlo Integrator)*
  4. **Impulse** *(Verlet-I/r-RESPA)*
  5. **ShadowHybridMC** *(Shadow Hybrid Monte Carlo Integrator)*
  6. **Umbrella**
- Single Timestep Integrators (STS)
  1. **BBK**
  2. **DMDLeapfrog** *(Self-consistent Leapfrog from Dissipative Particle Dynamics)*
  3. **DihedralHMC**
  4. **DihedralLiftMC**
  5. **LangevinImpulse**
  6. **Leapfrog** *(Velocity Leap-Frog Integrator)*
  7. **NPTVerlet**
  8. **NoseNVTLeapfrog**
  9. **PLeapfrog** *(Position Leap-Frog Integrator)*

10. **PaulTrap**

- Alias
  - HBondMOLLY : BSplineMOLLY

The choice of what integrator to use in ProtoMol is largely determined by the intended application. The main applications of molecular dynamics (MD) are equilibration, dynamics proper (e.g., to compute transport properties and autocorrelation functions), kinetics (e.g., to compute transition rates between metastable states), and sampling (e.g., to compute thermodynamic properties such as the free energy).

Computation of dynamics is more stringent than the other applications of MD. One should probably equilibrate a number of replicas from the desired ensemble (NVT or NPT) and then run NVE simulations. For NVE simulation, the standard integrator is the single time stepping Leapfrog. Usually the velocity version is preferred. However, if one uses multiple time stepping, the position version may be better. For solvated biomolecules the time step used in leapfrog is typically 1 fs, although the stability limit is around 2.25 fs. Using Shake (and optionally Rattle) to constrain bond lengths to Hydrogen allow longer time steps by roughly a factor of 2. Other tricks are used to use longer time steps, such as altering the mass of Hydrogens, etc.

Multiple time stepping (MTS) allows one to use longer time steps, although the presence of nonlinear and linear resonances (at one third and one half the fastest period in the system, cf. [9]) significantly limits the time steps available. The Verlet-I/r-RESPA/Impulse method can take long time steps of 3.3 fs without energy drift for very long times (several ns of simulation). One can get even longer steps using either the Equilibrium MOLLY [8] or the Bspline-MOLLY [5] method. Long time steps for unconstrained biomolecules of 6 fs have been reported, although care is recommended if using time steps of 5 fs or more. One should carefully analyze the time series of energy to see whether an intolerable drift is occurring over the length of the simulation.

For equilibration or sampling in NVT, we recommend using the Langevin Impulse method, which is exact for constant force. A good alternative is the self-consistent Leapfrog method (DMDLeapfrog), which preserves linear momentum and therefore hydrodynamics properties. We discourage the use of extended Hamiltonian methods such as Nose-Hoover (NoseNVTLeapfrog), since these are frequently not ergodic and have difficulty maintaining equipartition [4]. For equilibration in NPT, we have implemented the NPT Verlet method (NPTVerlet). This is probably the best equilibration scheme in ProtoMol, but it requires some fine tuning of the parameter scheme. Note that there are currently no minimization schemes in PROTOMOL. We recommend using NAMD 2.5 for minimization if required. This will be alleviated soon.

For sampling from the canonical (NVT) ensemble, one can also use Hybrid Monte Carlo methods to eliminate the systematic error due to discretization error in MD, which is manifested as time step dependence of the averages computed using MD. For small systems, Hybrid Monte Carlo is adequate. For larger systems (upward of hundreds of atoms), Shadow Hybrid Monte Carlo is the best choice. One has to choose a parameter c that controls efficiency and variance in observables, see [7] and [4].

Note that one can build a replica exchange protocol with ProtoMol - we currently do it using scripting, but this will be soon incorporated into the main code. This is a much more efficient way of sampling.

Integrator Argument Types

1. **BBK**

   timestep <length of step (float)>

   temperature <Kelvin temperature (float)>

   gamma <gamma (float)>

   seed <random seed (integer)>

2. **BSplineMOLLY**

   cyclelength <length of cycle (integer)>

   BSplineType <short|long>

   mollyStepSize <stepsize used for MOLLY integrator >

3. **DihedralHMC**

   cyclelength <Length of cycle (integer)>

   temperature <Kelvin temperature (in K)>

   randomCycLen <Use a random adjustment to cyclelength (boolean) >

   dihedralsSet <Flag to specify which dihedrals to move, if false then dihedrals chosen randomly (boolean)>

   dhmcDiSetFile <The dihedral indices are to specified in this file (string) >

   anglesSet <(boolean)>

   dhmcAnSetFile <(string) >

4. **DihedralLiftMC**

   cyclelength <Length of cycle (integer)>

   randomCycLen <Use a random adjustment to cyclelength (boolean) >

   temperature <Preferred system temperature (in K)>

5. **DMDLeapfrog**

   timestep <Size of each step (float)>

   iterations <number of iterations (integer)>

   gamma <gamma (float)>

   temperature <Kelvin temperature (float)>

   seed <seed to generate random numbers (integer)>

6. **EquilibriumMOLLY**

   cyclelength <length of cycle (integer)>

7. **HybridMC**

   cyclelength <length of cycle (integer)>

   randomCycLen <Use a random adjustment to cyclelength (bool)>

   temperature <Kelvin temperature (float)>

8. **Impulse**

   cyclelength <length of cycle (integer)>

9. **LangevinImpulse**

   timestep <length of step (float)>

   temperature <Kelvin temperature (float)>

   gamma <gamma (float)>

   seed <random seed (integer)>

10. **Leapfrog**

    timestep <length of step (float)>

11. **NoseNVTLeapfrog**

    timestep <length of each step (float)>

    temperature <preferred system temperature(in K)>

    thermal <heat bath coupling(1.0:very strong, 0:none) (float)>

    bathPos <history of the difference of system and heat bath (float)>

12. **NPTVerlet**

    timestep <length of step (float)>

    temperature <Kelvin temperature (in K) >

    pressure <(in Bar) >

    omegaTo <thermostat frequency>

    omegaTv <volume thermostat frequency >

    tauP <barostat time period >

13. **PaulTrap**

    timestep <length of step (float)>

    temperature <Kelvin temperature(in K)>

    thermal <(float)>

    bathPos <(float)>

    bathVel <(float)>

    scheme <thermostat scheme NVT|NVT_zero|NVT_ind| NVT_shell|NVT_global|berendsen|berendsen_zero |berendsen_ind| berendsen_shell|berendsen_global (string) >

    part <(float)>

    time <(vector)>

    t <(vector)>

14. **PLeapfrog**

    timestep <length of step (float)>

15. **ShadowHMC**

cyclelength <length of step (float)>

randomCycLen <Use a random adjustment to cycleLengthi (boolean) >

temperature <Preferred system temperature(in K)>

order <Desired order of approximation (4th or 8th)>

c <Parameter to specify divergence between shadow and total energy (float)>

16. **Umbrella**

    cyclelength <length of step (float)>

General Format of Integrator Forces

force <force1 type>
    <force1 arguments>
force <force2 type>
    <force2 arguments>
force <force3 type>
    <force3 arguments>
...

PROTOMOL supports primarily forces defined by the CHARMM forcefields (versions 19 and 27). There are some custom forces available (haptic force, friction, gravitation, Paul Trap, external field, external gravitation, Harmonic biasing force HarmDihedral, magnetic dipole, etc.) and it is easy to add new forces. You can look at the forces available in your copy of `ProtoMol` by running `protomol -f`.

You can include forces at will in your integrator. A nice tutorial on how to compose MTS integrators along with the forces is at `http://www.nd.edu/~izaguirr/papers/m3paper.pdf`. Some forces do not have parameters. This is particularly the case of bonded forces: angle, bond, dihedral, and improper. Nonbonded forces have several parameters, which fall primarily in these categories:

- **Potential:** Primarily Coulomb or Lennard Jones.
- **Algorithm:** This option refers to the algorithm used to compute the sum of pairwise interactions. Depending on the boundary condition and the potential, there are a number of algorithms:
  - For both Lennard Jones and Coulomb in periodic and vacuum boundary conditions, one can use cutoff computation (`-algorithm NonbondedCutoff`) or direct computational of all interactions (`-algorithm NonbondedFull`)
  - For Coulomb computation one can use an $O(N)$ multigrid summation (`-algorithm multigrid`) [6], both for vacuum or periodic boundary conditions. This is an attractive alternative to PME or Ewald, since it scales better in parallel. Setting the parameters requires some knowledge, so we recommend using the online recommender system MDSIMAID at http://mdsimaid.cse.nd.edu
  - For Coulomb one can also use PME (`-algorithm PMEwald`) or Ewald (`-algorithm FullEwald`).
- **Switching function:** when using cutoff or MTS integrators, one needs to bring the potential energy (and hence the force) smoothly to zero to avoid discontinuities that destabilize the integrators for MD. For LennardJones we recommend using a C^2 continuous switching function (`-switchingFunction`

`C2`). For Coulomb a C^1 switching function often suffices (`-switchingFunction C1`). It is also easy to add switching functions.

- **Method specific parameters:** Some methods require specific parameters, see the current force definitions accepted in `ProtoMol` by running `protomol -f`.

2.4 Required Parameters

As mentioned earlier, the command line must contain either the full pathname of the configuration file. These are the only restrictions specifically applied to the command line.

The following parameters MUST be specified on either the command line or in the configuration file specified on the command line (you may view Chapter 3 for a list of supported PROTOMOL files and their formats):

- One of either an initial positions file in PDB or XYZ or Binary format.
- One of an initial velocities file in PDB or XYZ or Binary format, or an initial temperature.
- A PSF topology file.
- A CHARMM Par parameters file.
- If the user specifies any specific output files that should be written, they must specify a filename.
- The number of steps for the simulation.
- A cubic cell manager and either vacuum or periodic boundary conditions.
- An integrator.

Chapter 3

PROTOMOL keyword and descriptions

3.1 Input files

keyword	type	Description
posfile	string	Contains the full or relative pathname of the initial positions file. PROTOMOL supports PDB, XYZ or Binary formatted position files.
velfile	string	Contains the full or relative pathname of the initial velocities file. Once again, PROTOMOL supports PDB, XYZ or Binary formats. If no initial velocity file is specified, random velocities are generated based on the initial temperature and a seed that can also be specified. Therefore one of either the velfile, or temperature needs to be specified. A seed is optional.
psffile	string	Contains the full or relative pathname of the initial topology file in PSF format.
parfile / parameters	string	Contains the full or relative pathname of the initial CHARMM parameter file.

3.2 Boundary Conditions

keyword	type	Description
boundaryconditions	string	PROTOMOL supports vacuum or periodic boundary conditions, specified with “Vacuum” or “Periodic”, respectively.
cellbasisvector1	x y z (x, y, and z are floats)	Basis vector 1, for periodic boundaries
cellbasisvector2	x y z (x, y, and z are floats)	Basis vector 2, for periodic boundaries

cellbasisvector3	x y z (x, y, and z are floats)	Basis vector 3, for periodic boundaries
cellorigin	x y z (x, y, and z are floats)	Center of periodic cell, for periodic boundaries

3.3 Output Files

keyword	type	Description
dcdfile	string	Contains the full or relative pathname of the DCD trajectory file to be written, if desired.
dodcdfile	boolean	Specifies if the user would like a DCD trajectory file to be written.
allenergiesfile	string	Contains the full or relative pathname of the energies file to be written, again if desired.
doallenergiesfile	boolean (default: true)	Specifies if the user would like an energies file to be written, with all energies (bond, angle, dihedral... whatever was forced in the integrator section) in one file.
allenergiesFileOutputFreq	integer (default: outputFreq)	Specifies the frequency of the energies file to be written, if desired.
allEnergiesFileCacheFreq	integer (default: 1)	Specifies frequency in number of lines to flush the cache to file.
allEnergiesFileCacheSize	integer (default: 0)	Specifies the minimal size of cached data to flush to file
allEnergiesFileCloseTime	float (default: 1.0)	Specifies the minimal time interval between two writes to close the file temporarily
outputfreq	integer	Specifies the frequency in timesteps for the writing of energy data to the console and defines the default frequency for other outputs.
finpdbposfile	string	Contains the full or relative pathname of the final positions file in PDB format, if the user desires.
dofinpdbposfile	boolean	Specifies if the user would like a final PDB positions file to be written.
xyzposfile	string	Contains the full or relative pathname of the positions trajectory file in XYZ format, once again if desired.
doxyzposfile	boolean	Specifies if the user would like an XYZ trajectory file for positions to be written.
xyzposFileOutputFreq	integer (default: outputFreq)	Specifies the frequency of the XYZ trajectory file for positions to be written, if desired.

xyzvelfile	string	Contains the full or relative pathname of the XYZ velocities trajectory file, if desired.
doxyzvelfile	boolean	Specifies if the user would like an XYZ velocities trajectory file to be written.
xyzvelFileOutputFreq	integer (default: outputFreq)	Specifies the frequency of the XYZ trajectory file for velocities to be written, if desired.
momentumfile	string	Contains the full or relative pathname of the momentum output file to be written, if desired. The output contains the momentum for each dimension..
domomentumfile	boolean	Specifies if the user would like a momentum output file to be written.
momentumFileOutputFreq	integer (default: outputFreq)	Specifies the frequency of the tmomentum file to be written, if desired.
MomentumFileCacheFreq	integer (default: 1)	Specifies frequency in number of lines to flush the cache to file.
MomentumFileCacheSize	integer (default: 0)	Specifies the minimal size of cached data to flush to file
MomentumFileCloseTime	float (default: 1.0)	Specifies the minimal time interval between two writes to close the file temporarily
finXYZBinPosFile	string	Specifies the name of the output file which will contain the co-ordinates are recorded at the end of the simulation in XYZ binary format.
dofinXYZBinPosFile	boolean	Flag which specifies whether to generate final positions in XYZ binary format
finXYZBinVelFile	string	Specifies the name of the output file which will contain the velocities are recorded at the end of the simulation in XYZ binary format.
dofinXYZBinVelFile	boolean	Flag which specifies whether to generate final veclocities in XYZ binary format
finXYZPosFile	string	Specifies the name of the output file which will contain the co-ordinates are recorded at the end of the simulation in XYZ(ASCII) format.
dofinXYZPosFile	boolean	Flag which specifies whether to generate final positions in XYZ(ASCII) format
finXYZVelFile	string	Specifies the name of the output file which will contain the velocities are recorded at the end of the simulation in XYZ(ASCII) format.
dofinXYZVelFile	boolean	Flag which specifies whether to generate final veclocities in XYZ(ASCII) format

paulfile	string	Contains the full or relative pathname of the Paul trap output file to be written, if desired. The output contains the kinetic energy, temperature, energy difference of Coulomb minus twice the Paul trap, and a histogram in function of the distance (position) to the origin.
dopaulfile	boolean	Specifies if the user would like a Paul trap output file to be written.
paulFileOutputFreq	integer (default: outputFreq)	Specifies the frequency of the Paul trap output file to be written, if desired.
PaulFileCacheFreq	integer (default: 1)	Specifies frequency in number of lines to flush the cache to file.
PaulFileCacheSize	integer (default: 0)	Specifies the minimal size of cached data to flush to file
PaulFileCloseTime	float (default: 1.0)	Specifies the minimal time interval between two writes to close the file temporarily
paulLowFile	string	Specifies the filename where minimal positions and paulTrap output are recorded
doPaulLowFile	boolean	Used to set or reset paulTrap minimal output and positions
diffusionFile	string	File to print the diffusion coefficient computed up to each time step
dodiffusionFile	boolean	Flag to specify whether to output diffusion coefficient computed up to each time step
diffusionFileOutputFreq	integer	Used to set output frequency of diffusion coefficient
DiffusionFileCacheFreq	integer (default: 1)	Specifies frequency in number of lines to flush the cache to file.
DiffusionFileCacheSize	integer (default: 0)	Specifies the minimal size of cached data to flush to file
DiffusionFileCloseTime	float (default: 1.0)	Specifies the minimal time interval between two writes to close the file temporarily

3.4 Output dihedrals

keyword	type	Description
dihedralsFile	string	Specifies the filename where the dihedral angle values will be stored
dodihedralsFile	boolean	This flag specifies whether to output dihedrals in the dihedralsFile
dihedralsIndex	Integer	Specifies the index of the dihedral in the psf file for which the output have to be recoreded in dihedralsFile.
dihedralsFileOutputFreq	Integer	Specifies the frequency of output for dihedrals (e.g. a frequency of 1 implies the dihedral angle is recorded at every step of the integration.)
DihedralsFileCacheFreq	integer (default: 1)	Specifies frequency in number of lines to flush the cache to file.
DihedralsFileCacheSize	integer (default: 0)	Specifies the minimal size of cached data to flush to file
DihedralsFileCloseTime	float (default: 1.0)	Specifies the minimal time interval between two writes to close the file temporarily
dihedralsSet	boolean	It should be set to true if you want to record multiple dihedrals.
dihedralsSetFile	string	Specifies the filename where one can specify multiple dihedral indices, one in each line

3.5 Parallel Mode

keyword	type	Description
parallelpipe	integer (default: 0)	Specifies if the depth of the pipe to assign work to each slave.
usebarrier	boolean (default: true)	Specifies if explicit synchronization (barrier) is desired before global communication.
maxPackages	integer	Maximum number of work packages per node per force; increased packages, better load balance, but more communication(Default is -1)
parallelMode	string	static, dynamic or masterSlave, where static : static load balancing, no com. between master and slaves, only slaves, and dynamic : master-slave, where the master does some work inbetween

3.6 Simulation Setup

keyword	type	Description
numsteps	integer	Specifies the number of steps for the simulation to run.
firststep	integer (default: 0)	Specifies the number of the initial timestep.
temperature	float	Specifies the initial Kelvin temperature.
seed	unsigned integer	Random number seed for velocity generation. This is only used if an initial velocity is not specified, and in that case if the seed is specified as 0, it defaults to the timer.
cellmanager	string	Cell manager - current PROTOMOL only supports a cubic cell manager, specified by "Cubic". The reason why it needs to be specified even though there is only one option is to allow for future flexibility.
cellsize	float	Specifies the size of cell used by the cell manager.
exclude	"none", "1-2", "1-3", "1-4", "scaled1-4" (default: "scaled1-4")	Specifies nonbonded exclusions
integrator	string	Specifies the name of integrator(s) to be used
shadowEnergy	boolean	Specifies whether to compute shadow hamiltonian
paulOmega	real	
paulOmegaZ	real	
Screen	boolean	Output to the screen
ScreenOutputFreq	integer	Used to set frequency of screen output
ScreenPaulTrap	boolean	Used to set screen output frequency for PaulTrap

3.7 Parameters for RATTLE and SHAKE

keyword	type	Description
rattle	boolean	Use this flag to constrain velocities
rattleEpsilon	real	Error tolerance for rattle (default=$1e-5$)
rattleMaxIter	integer	Maximum number of iterations for rattle (default=30)
shake	boolean	Use this flag to constrain positions
shakeEpsilon	real	Error tolerance for shake (default=$1e-5$)
shakeMaxIter	integer	Maximum number of iterations for shake (default=30)

3.8 Miscellaneous

keyword	type	Description
virialCalc	boolean	Compute virial (default=false)
molVirialCalc	boolean	Compute molecular virial (default=false)
molecularTemperature	boolean	compute molecular temperature (default=false)
energiesFileOutputFreq	boolean	
coulombScalingFactor	real	Scaling factor for Coulomb interactions when using scaled1-4 exclusions (default=1, appropriate for CHARMM)

Chapter 4

PROTOMOL Configuration Files and Test Molecule Examples

The PROTOMOL source release includes a folder of example simulation configurations and select test molecules. Examples are organized by test molecule and each folder contains one or more simulation configuration files. To provide the user with a uniform set of example simulation configurations, the solvated BPTI test molecule with 14281 atoms includes the most robust set of configuration files. Each test molecule example includes a README file which describes the origination of the molecular parameters and any preprocessing (equilibration, minimization, etc...) which has been performed. The following three subsections provide the reader with a more indepth look at representative simulation configuration files.

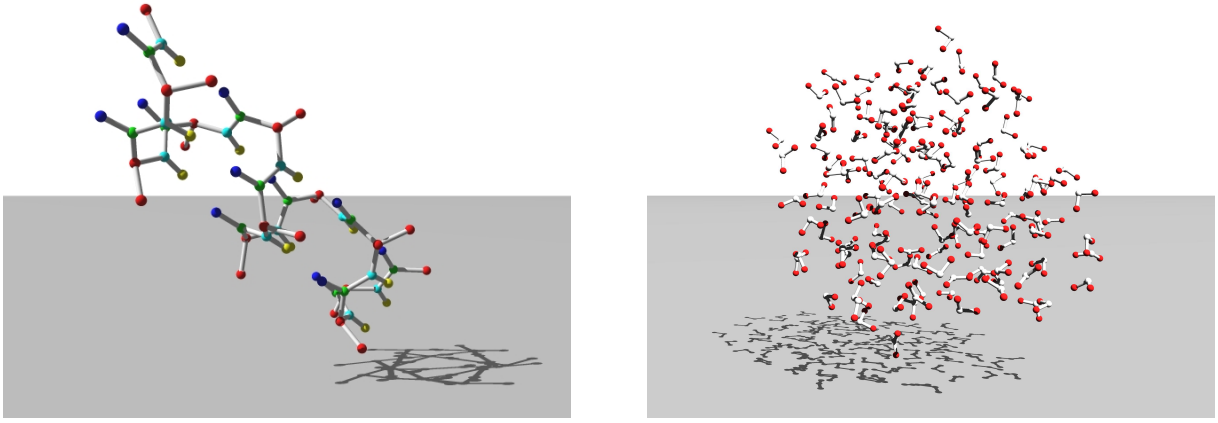

Figure 4.1: **Left:** alanin 66 , **right:** 10 Å diameter water droplet.

4.1 Alanin Configuration File with Leapfrog Integrator

Following is an example of a configuration file for the protein molecule alanin (Fig.4.1) . In this case, PROTOMOL will run 10000 steps starting at time 0, and random velocities will be generated based on the initial temperature of 300.0 and the random seed of 1234. A cubic cell manager is used (this must be true) with a cell size of 6.5. Output files, in this case just an all energies file and a DCD trajectory file will be written every 100 timesteps. Also a final PDB position and velocity file will be written. The initial parameter file alanin.par uses the newer CHARMM format. Vacuum boundary conditions are present, and

the integrator is a single-timestep leapfrog integrator, with all bonded and two nonbonded forces present. The van der Waals and electrostatic forces both use a nonbonded cutoff algorithm and have cutoffs at 6.5. The switching function for the van der Waals potential function makes the second derivative continuous at the cutoff point, while the electrostatic potential function has a continuous first derivative due to its switching function. The switchon option has been used for van der Waals (recall that it can only be used with the C2 switching function).

```
temperature 300.0
firststep 0
numsteps 10000
cellsize 6.5
outputfreq 100
seed 1234
posfile alanin.pdb
psffile alanin.psf
parfile alanin.par
finpdbposfile alanin.out.pos.pdb
finXYZvelfile alanin.out.vel.pdb
DCDfile alanin.out.dcd
allenergiesfile alanin.out.energy
boundaryConditions vacuum
cellManager Cubic
Integrator {
    level 0 Leapfrog {
             timestep 1
      force Improper
      force Dihedral
      force Bond
      force Angle
      force LennardJones
               -algorithm NonbondedCutoff
               -switchingFunction C2
        -switchon 1.0
               -cutoff 12
      force Coulomb
               -algorithm NonbondedCutoff
               -switchingFunction C1
               -cutoff 12
    }
}
```

4.2 20 Å diameter water droplet With A Two-Level Integration Scheme

This sample of water (Fig.4.1) is being run using a two-level integration scheme, starting with an MTS Impulse Integrator with a cycle length of 5 and finishing with an STS LeapFrog Integrator that has a timestep of 1.0. The initial temperature is 300 and the first timestep is 0, and the simulation will be run for 50 steps. Note the cubic cell manager and periodic boundary conditions, and that all cell basis vectors have been provided. Output files, however, will be written, including a DCD trajectory file and an all energies file, every 10 steps. Note that all necessary initial data files have been provided. Once again, for the level 1 integrator the Coulombic interaction is computed using Ewald [2, 3] summation.

```
temperature 300.0
firststep 0

numsteps 50

cellsize 6.5

outputfreq 10
seed 1234

posfile equil298K_01.pos.pdb
velfile equil298K_01.vel.pdb

psffile equil298K_01.psf
parfile equil298K_01.par

ALLENERGIESFILE   equil298K_01.out.energies

cellBasisVector1     25.0 0.0 0.0
cellBasisVector2     0.0 25.0 0.0
cellBasisVector3     0.0 0.0 25.0
cellorigin           0.0 0.0  0.0

boundaryConditions Periodic
cellManager Cubic

Integrator {

  level 1 Impulse {
        cyclelength 4
    force Coulomb -algorithm FullEwald -reciprocal
  }
  level 0 Leapfrog {
        timestep 1.0
    force Coulomb -algorithm FullEwald -correction -real
    force Bond, Angle
    force LennardJones
          -algorithm NonbondedCutoff
          -switchingFunction C2
          -cutoff 6.5
      -switchon 0.1
  }
}
```

4.3 BPTI with Hybrid Monte Carlo Sampling

Here is an example of the Hybrid Monte Carlo algorithm used to sample a BPTI molecule - a 2-level Hybrid Monte Carlo MTS integrator is used. Once again, the initial timestep is at time 0, but the number of timesteps to run this time is 100. 1-2 exclusions are desired, and the cellsize is once again 6.5. Initial temperature has been specified to be 300 to initialize with random velocities. There are now however periodic boundary conditions, and as a result three cell basis vectors along with a cell origin are required. Output files will be

written to every 10 timesteps - this time they include XYZ position and trajectory files, a DCD trajectory file, and energies file (thus there will be 10 different energy files generated - one for each type of energy). Note that a final XYZ position file will be written since dofinxyzvelfile is set to “yes”. Since commotion is set to “yes”, center of mass motion will be removed when calculating velocities. Once again notice the cubic cell manager, and also notice that the HMCIntegrator does not have forces. The integrator will run 5 Hybrid Monte Carlo cycles, with a cycle length of 10 and 50 warm-up cycles, at a temperature of 300 K. Four bonded forces (bond, angle, improper, dihedral) are present, and one nonbonded (FullEwald, the Coulomb force solved for using Ewald summation).

```
temperature 300
 firststep 0
 numsteps 100
 exclude 1-2
 cellsize 6.5
 cellbasisvector1  32.0    0    0
 cellbasisvector2     0 32.0    0
 cellbasisvector3     0    0 32.0
 cellorigin           0    0    0
 outputfreq 10
 posfile ../data/bpti.pdb
 psffile ../data/bpti.psf
 parfile ../data/bpti.par
 finxyzposfile bpti.out.pos.xyz
 finxyzvelfile bpti.out.vel.xyz
 dofinxyzvelfile yes
 xyzposfile bpti.out.trajectory.pos.xyz
 dcdfile bpti.out.dcd
 allenergiesfile bpti.out.energy
 boundaryConditions Periodic
 commotion yes
 cellManager Cubic
 Integrator {
       level 1 HybridMC {
               cyclelength 10
               warmupcycles 50
               temperature 300.0
    }
      level 0 PLeapFrog {
               timestep 1
       force Bond, Angle, Dihedral, Improper
       force Coulomb
          -algorithm FullEwald
          -real
          -reciprocal
          -correction
    }
 }
```

4.4 Shadow Hybrid Monte Carlo

Here is an example of Shadow Hybrid Monte Carlo(SHMC) algorithm using the BPTI molecule-a 2 level ShadowHMC integrator is used. The acceptance rate of Hybrid Monte Carlo techniqye decreases exponentiall with increasing timestep δt and N, the system size. SHMC is a generalizations of HMC technique that samples form a p. d. f. in all phase space. It achieves an asymptotic speedup of $O(N^{\frac{1}{4}})$ over Hybrid Monte Carlo. The initial timestep is 0 and the total number of timesteps to simulate is 10. ShadowHMC takes as input 4 parameters. The *temperature* is the simulation temperature. The seed initializes the random number generator. Like HMC, SHMC uses random number generator drand(). SHMC needs a tuning parameter c to indicate the divergence between shadow and total energy. SHMC requires the integrator to be symplectic in order to compute shadow hamiltonian. That is Verlet/Leapfrog is used as innner integrator. The cyclelength specifies the number of steps before the energy difference is computed and the metropolis criteria is checked. Here it is 25. The order specifies the order of the accuracy of the shadow hamiltonian. In PROTOMOL , we have only 2 options are available for order, namely 4 and 8. Please see [7] for further details.

```
debug 0
numsteps 10
firststep 0

seed 7536031

posfile bpti.pdb
psffile bpti.psf
parfile bpti.par
temperature 300

outputfreq 1

allenergiesfile bpti.out.energies.shmc

boundaryConditions Periodic
# cellBasisVector1 64.32 0 0
# cellBasisVector2 0 51.167 0
# cellBasisVector3 0 0 51.272

cellManager Cubic
cellsize 4

# removeLinearMomentum  1
# removeAngularMomentum 1
shadowEnergy true


Integrator {

    level 1 ShadowHMC {

        temperature 300
        cyclelength 25
        order 8
        c 0
```

```
    }
    level 0 Leapfrog {

        timestep 0.5

        force Improper
        force Dihedral
        force Bond
        force Angle

        force Coulomb
             -algorithm NonbondedSimpleFull

        force LennardJones
             -algorithm NonbondedSimpleFull

    }
}
```

4.5 Langevin Impulse

Here is an example of Langevin Impulse integrator [11] using united-atom butane. The temperature is the target temperature of the molecule/subsystem. Gamma specifies the collision parameter or the damping constant. The *N*onbondedSimpleFull option specifies that the direct method is used to compute non-bonded interactions with no cutoff. This works only for non-periodic boundary conditions. PROTOMOL 2.0.3 comes with a feature where one can output the dihedral angle values in a file. The phrase dihedralsfile specifies the filename where dihedral angle values are recorded in radians. Of course , you'll have to set the dodihedralsfile flag to make it work. dihedralsIndex specifies the index of the dihedral. This index specifies the order of the dihedral angle in psf file which you are interested in. In this example, we are using United-atom butane and there is only one dihedral, with index 0. To record multiple torsion angle one can use the dihedralsSetfile flag to specify an input filename where one can list the dihedral indices, one in each line. Note that dihedralsSet flag should be set in order for PROTOMOL to take multiple dihedral indices as input.

```
temperature 300
firststep 0

numsteps 500

cellsize 5

outputfreq 10

posfile UA_butane.pdb
psffile UA_butane.psf
parfile UA_butane.par

allenergiesfile ua_butane.lang.out.energy

boundaryConditions vacuum

cellManager Cubic
```

```
dodihedralsfile true
dihedralFileOutputFreq 1
dihedralsfile ua_butane.lang.out.dihedrals
dihedralsIndex 0
dihedralsSet false
dihedralsSetfile testset18

Integrator {
  level 0 LangevinImpulse {
    timestep 1.0
    temperature 300
    gamma 7000
    seed 0
    force Bond
    force Angle
    force Dihedral
    force Improper
    force LennardJones Coulomb
      -algorithm NonbondedSimpleFull
  }
}
```

4.6 Interaction of PROTOMOL with VMD

You may start VMD normally before you begin running the desired simulation to be viewed in PROTOMOL. Select the "molecule" option from the main window, and in the resulting molecule window select "Load From Files". Then in the files window that opens, under the menu "Molecule File Types" select psf and pdb, then write the full pathnames of the PSF and PDB files used in the simulation that will be run in the provided spaces. Now VMD has the initial state of the molecule to be simulated.

Next, in the configuration file that will be used, make sure there is a Haptic force in the integrator. Check to see that a positive nonzero value is supplied for the "-wait" value under the haptic force. This will force PROTOMOL to wait some time for an IMD connection, 100 would be a decent value so that you have time to set VMD to simulate. Also note the port number, which will be used later.

Now run PROTOMOL. It should display "Waiting for IMD connection....." after a short time. At this point, go to the VMD console window and type "`imd connect <hostname> <port number>`", where `<hostname>` is the name of the machine that PROTOMOL is running on and `<port number>` is the port number value specified in the configuration file that is being used.

At this point, VMD will view the molecule throughout the simulation as PROTOMOL runs. Also from the VMD window IMD commands can be run that will send messages to PROTOMOL. For example, typing "`imd trate <integer value>`" will change the transmission rate for information from PROTOMOL to VMD.

As an example, let us return to the integrator section of the alanin configuration file from section 4.1:

```
Integrator {
```

```
      level 0 Leapfrog {
               timestep 1
        force Improper
        force Dihedral
        force Bond
        force Angle
        force LennardJones
                 -algorithm NonbondedCutoff
                 -switchingFunction C2
          -switchon 0.1
                 -cutoff 12
        force Coulomb
                 -algorithm NonbondedCutoff
                 -switchingFunction C1
                 -cutoff 12
        force Haptic
          -port 2001
          -trate 1
          -timeout 1000
          -step_inc 100
          -wait 100
      }
  }
```

The only area that counts as far as the VMD simulation is concerned is the "`force Haptic`" section at the end. Notice first that a `-wait` value is given that is greater than zero, thus PROTOMOL will wait for an IMD connection, which is what we want. The port ID is 2001. The other three values are IMD variables.

In this case as soon as PROTOMOL begins waiting for the IMD connection, a user would type "`imd connect <name of the machine on which ProtoMol is running> 2001`" and the visual simulation will begin. Now the user can run IMD commands in the IMD window while the simulation is running.

Figure 4.2: PHANTOM Desktop^{TM} Haptic Device.

Appendix A

Contact Information

The best contact path for licensing issues is by e-mail to *protomol@cse.nd.edu* or send correspondence to:

PROTOMOL Team
c/o Prof. Jesús A. Izaguirre
Laboratory for Computational Life Sciences
Department of Computer Science and Engineering
University of Notre Dame
384 Fitzpatrick Hall of Engineering
Notre Dame, Indiana 46556 USA

Bibliography

[1] M. Bhandarkar, R. Brunner, A. Dalke, J. Gullingsrud, A. Gursoy, W. Humphrey, D. Hurwitz, N. Krawetz, M. Nelson, J. Phillips, A. Shinozaki, G. Zheng, and F. Zhu. Namd user's guide: Version 2.3b2. page 11, 405 N. Mathews, Urbana, IL 61801, 2001. Theoretical Biophysics Group at the University of Illinos and Beckman Institute.

[2] S. W. de Leeuw, J. W. Perram, and E. R. Smith. Simulation of electrostatic systems in periodic boundary conditions. I. Lattice sums and dielectric constants. *Proc. R. Soc. Lond. A*, 373:27–56, 1980.

[3] P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Ann. Phys.*, 64:253–287, 1921.

[4] S. Hampton, P. Brenner, A. Wenger, S. Chatterjee, and J. A. Izaguirre. Biomolecular sampling: Algorithms, test molecules, and metrics. To appear in *Proc. of Algorithms for Macromolecular Modeling IV*. Lecture Notes in Computational Science and Engineering (LNCSE). Springer Verlag. New York and Berlin. Manuscript available at `http://www.nd.edu/~izaguirr/papers/samp-am3.pdf`, 2005.

[5] J. A. Izaguirre. *Longer Time Steps for Molecular Dynamics*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1999.

[6] J. A. Izaguirre, S. Hampton, and T. Matthey. Parallel multigrid summation for the N-body problem. *J. Paral. Distrib. Comp.*, 2005. In press. Manuscript available at `http://www.nd.edu/~izaguirr/papers/MaIz0x_JPDC.pdf`.

[7] J. A. Izaguirre and S. S. Hampton. Shadow hybrid Monte Carlo: An efficient propagator in phase space of macromolecules. *J. Comput. Phys.*, 200(2):581–604, 2004.

[8] J. A. Izaguirre, S. Reich, and R. D. Skeel. Longer time steps for molecular dynamics. *J. Chem. Phys.*, 110(19):9853–9864, 1999.

[9] Q. Ma, J. A. Izaguirre, and R. D. Skeel. Verlet-I/r-RESPA/Impulse is limited by nonlinear instability. *SIAM J. Sci. Comput.*, 24(6):1951–1973, 2003.

[10] T. Matthey, T. Cickovski, S. S. Hampton, A. Ko, Q. Ma, M. Nyerges, T. Raeder, T. Slabach, and J. A. Izaguirre. PROTOMOL: An object-oriented framework for prototyping novel algorithms for molecular dynamics. *ACM Trans. Math. Softw.*, 30(3):237–265, 2004.

[11] R. D. Skeel and J. A. Izaguirre. An impulse integrator for Langevin dynamics. *Mol. Phys.*, 100(24):3885–3891, 2002.