GENESIS User Guide

Release 1.0

RIKEN AICS

2015/05/08

GENESIS release 1.0

Project Leader: Yuji Sugita (RIKEN)

Main Developers: Jaewoon Jung (RIKEN), Takaharu Mori (RIKEN)

Developers: Chigusa Kobayashi (RIKEN), Yasuhiro Matsunaga (RIKEN), Takashi Imai (RIKEN), Takao Yoda (Nagahama Institute of Bio-Science and Technology)

Contributors: Naoyuki Miyashita (RIKEN), Ryuhei Harada (RIKEN), Wataru Nishima (RIKEN), Raimondas Galvelis (RIKEN), Yasuaki Komuro (RIKEN)

Acknowledgments: Norio Takase (Isogo Soft), Hikaru Inoue (Fujitsu), Tomoyuki Noda (Fujitsu)

Citation Information

J. Jung, T. Mori, C. Kobayashi, Y. Matsunaga, T. Yoda, M. Feig, and Y. Sugita, "GENESIS: A hybrid-parallel and multi-scale molecular dynamics simulator with enhanced sampling algorithms for biomolecular and cellular simulations", WIREs Computational Molecular Science (DOI: 10.1002/wcms.1220)

Copyright Notice

GENESIS is distributed under the GNU General Public License version 2.

It should be mentioned this package contains the following softwares for convenience. Please note that these are not covered by the license under which a copy of GENESIS is licensed to you, while neither composition nor distribution of any derivative work of GENESIS with these software violates the terms of each license, provided that it meets every condition of the respective licenses.

Mersenne Twister: A random number generator

A Mersenne Twister random number generator was originally written in C by Makoto Matsumoto and Takuji Nishimura, and later translated into Fortran by Hiroshi Takano and Richard Woloshyn. This routine is distributed under the GNU General Public License version 2.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details. You should have received a copy of the GNU Library General Public License along with this library; if not, write to the Free Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

FFTE: A Fast Fourier Transform Package

FFTE (http://www.ffte.jp/) is written by Daisuke Takahashi (Tsukuba University).

Complementary error function: erfc04

A Complementary error function routine (erfc04) is written by SunSoft, a Sun Microsystems, Inc. business.

Note:

"GENESIS (General Neuron Simulation System)" (http://www.genesis-sim.org/) simulator, which has much longer history in the development.

[The Book of GENESIS]

`http://www.amazon.com/`
`The-Book-GENESIS-Exploring-SImulation/dp/0387949380/`
`ref=sr_1_1?ie=UTF8&qid=1394425468&sr=8-1&keywords=bower+beeman`

Our "GENESIS" has been developed independently to investigate conformational dynamics of proteins, nucleic acids, biological membranes, and other biomolecules.

CONTENTS

CHAPTER

ONE

INTRODUCTION

GENESIS (*Gen*eralized-*E*nsemble *Si*mulation *S*ystem) is a suite of computer programs for carrying out molecular dynamics (MD) simulations of biomolecular systems. MD simulations of biomolecules such as proteins, nucleic acids, lipid bilayers, N-glycans, and so on, are used as important research tools in structural and molecular biology. Many useful MD simulation packages [1] [2] [3] [4] [5] are now available together with accurate molecular force field parameter sets [6] [7] [8] [9] [10]. Most of the MD software have been optimized and parallelized for distributed-memory parallel supercomputers or PC-clusters. Therefore we can use hundreds of CPUs or CPU cores efficiently for a single MD simulation of a relatively large biomolecular system, typically composed of several hundreds thousand of atoms. In recent years, the number of available CPUs or CPU cores is rapidly increasing, and thereby, modern and more efficient parallel schemes are necessary to be implemented in the MD simulation programs.

One of our major motivations is to develop MD simulation software whose performance is scalable on such modern supercomputers. For this purpose, we have developed the software from scratch, introducing the hybrid (MPI + OpenMP) parallelism and several new parallel algorithms [11] [12]. Another major motivation is to develop a MD simulation code, which can be easily understood and modified for methodological development. The two policies (high parallel performance and simplicity) usually conflict with each other in computer software. Therefore we considered to develop two MD simulators simultaneously.

They are **SPDYN** (*Sp*atical decomposition *dyn*amics) and **ATDYN** (*At*omic decomposition *dyn*amics). Although these two MD codes share almost the same data structures, subroutines, and modules, a different parallelization scheme is introduced in each simulator. In **SPDYN**, the spatial decomposition scheme is implemented with new parallel algorithms [11] [12]. Its performance is therefore better than **ATDYN** and most of other previously developed MD simulators. In **ATDYN**, the atomic decomposition scheme is introduced aiming for simplicity in the source code; and enhanced conformational sampling algorithms such as the replica-exchange molecular dynamics (REMD) method is available.

Due to the simple parallelization, the performance of **ATDYN** is worse than **SPDYN**. However, **ATDYN** is simpler and, thereby, easier to modify for development of new algorithms or novel molecular models. We hope, ambitious users will try to develop new methodologies in **ATDYN** at first and, eventually, move to **SPDYN** for the better performance. As we try to maintain consistency between the source codes of **ATDYN** and **SPDYN**, switching from **ATDYN** to **SPDYN** is possible by ambitious users.

Other features in **GENESIS** are listed as follows:

- Not only atomistic molecular force field (CHARMM) but also Go-model is available in **ATDYN**.
- For extremely large biomolecular systems (more than 10 million atoms), parallel input/output (I/O) scheme is implemented and available.
- **GENESIS** is optimized for "K computer" (developed by RIKEN and Fujitsu company), but it is available on Intel-based supercomputers or PC-clusters.

- **GENESIS** is written in modern Fortran 90/95/2003 using modules and dynamic memory allocation. No common blocks are used!
- **GENESIS** is free software under the GNU General Public License (GPL) version 2 or later. We allow any uses to use/modify **GENESIS** and redistribute the modified version under the same license.

And so on.

This manual consists of 18 chapters including information how to get started, explanation of each keyword in control files, and tutorials for MD simulations, coarse-grained (CG) MD simulations with Go-model, REMD simulations, and so on. We recommend new users of **GENESIS** to start from the next chapter, *Getting Started*, to learn a general idea, installation, and work flow of the program.

Compared to other MD software like AMBER, CHARMM, NAMD, and so on, **GENESIS** is a very young MD simulator. Before releasing the program, the developers and contributors in **GENESIS** development team worked hard to kill all bugs in the program, and performed a bunch of test simulations. Still, there is possibility of several defects or minor bugs in **GENESIS**. Since we cannot bear any responsibility for the simulation results produced by **GENESIS**, we recommend new users to check their results carefully, if necessary, by comparing with other MD programs.

We, **GENESIS** development team, have a lot of plans for future development of methodology and molecular models. Some of them are already on-going projects by us. We would like to grow **GENESIS** toward one of the most powerful and feasible MD software package, contributing to computational chemistry and biophysics. We believe that the current status of computational studies in life science area is still in the very early stage (like 'GENESIS') compared to established experimental researches. We hope, **GENESIS** can push forward the computational science and contribute to life-scientific and medical applications in the near future.

CHAPTER

TWO

GETTING STARTED

GENESIS consists of two simulators and several analysis tools. The simulators, called **ATDYN** and **SPDYN**, can perform minimization, molecular dynamics, and other advanced simulations of biomolecules. The analysis tools, **trj_analysis**, **crd_convert**, **pcrd_convert**, **remd_convert**, are used for post-processing trajectories produced by the simulators. **prst_setup** generates **GENESIS** original multiple restart I/O data from the conventional CHARMM input data.

A description of each program is given in the next chapter (*Available Programs*), and a detailed usage (including references for input parameters) is explained in *Tutorials* (from Chapter 14). This chapter is devoted to orient new users who have just downloaded **GENESIS** package. In the first half of this chapter, compilation and installation of **GENESIS** is described. In the later half, we give the users a general idea of how to use **GENESIS** for their own purposes.

2.1 How to install GENESIS

2.1.1 Requirements

The source code of **GENESIS** is written in Fortran 90/95/2003 and organized into *modules*. The preprocessing directives (such as `#ifdef`) are used to adapt the source code for different architectures and options. Fortran compiler and its preprocessor is the minimum requirement to compile **GENESIS**.

The simulators (**ATDYN** and **SPDYN**) of **GENESIS** are highly parallelized in their own decomposition schemes (atomic decomposition and spatial decomposition, respectively). These schemes are implemented by using both of *de facto* standard parallel programming models: MPI (distributed-memory) and OpenMP (shared-memory). In a nutshell, MPI is used for the communication between the different machines (nodes) or processes, while OpenMP is used within a single machine (node) or process. Whereas OpenMP is supported by many of modern Fortran compilers, MPI needs to be installed in your environment if necessary. In particular, **SPDYN** is designed for large-scale parallel application.

Although **GENESIS** is tested on various platforms, at present, validated combinations of CPU and compiler are limited to those provided by Intel and Fujitsu. Thus, the combinations of Intel CPU and compiler, or SPARC CPU and Fujitsu compiler is recommended. As for the MPI library, we have thoroughly tested with OpenMPI [13], which is another recommendation.

The recommendations for compiling **GENESIS** are summarized below. Please make sure that at least one of them in each section is installed on your system.

- Fortran compiler
  - Intel compiler `ifort` (Recommended)

  - Fujitsu compiler `frtpx` (Recommended)
  - GCC compiler `gfortran` (4.4.7 or higher version required; some features do not work)
- Preprocessor
  - `fpp` supplied with Intel compiler (Recommended)
  - Fujitsu compiler `frtpx` supports preprocessing (Recommended)
  - GCC preprocessor `cpp`
- MPI implementation
  - OpenMPI (Recommended)
  - Fujitsu MPI (Recommended)
  - Intel MPI (Recommended)
  - MPICH (Tested, but not recommended)
- Operating system
  - Linux (Recommended)
  - Mac OSX (Not recommended, but may work)

2.1.2 Download

GENESIS package is available at http://www.riken.jp/TMS2012/cbp/en/research/software/genesis/index.html
The files are organized according to their purpose.

- User Guide (`genesis.pdf`): this file
- Source code (`genesis.tgz`): source code for the simulators and the tools
- Test files (`test.tgz`): script and input files for regression tests (see below)
- Tutorials (`tutorial.tgz`): input files for tutorials, see *Tutorials* (from Chapter 14)

2.1.3 Installation

First, extract the archive of the source code (`genesis.tgz`).

```
# untar the package file and change the working directory
$ tar xvfz genesis.tgz
$ cd genesis/
$ ls -lF
total 8
-rw-r--r--   1 user  staff   79  9 12 22:27 README  # README file
-rw-r--r--   1 user  staff   79  9 12 22:27 COPYING # License agreement
drwxr-xr-x  10 user  staff  340 10 20 18:16 src/    # Source codes
```

GENESIS uses GNU autotools build system. Change the current directory to `src/`, and run `./configure` script to create a `Makefile` for your system. `./configure` script tries to detect all of the requirements needed to compile **GENESIS**:

```
# change the working directory to src/ and invoke ./configure
$ cd src/
$ ./configure
```

It may take a while `./configure` script to complete. While running, it prints messages telling which things are checked.

If you need to add compilation options and/or library paths, you can set the following options:

options	**meaning**
–prefix=PREFIX	install architecture-independent files in PREFIX
–exec-prefix=EPREFIX	install architecture-dependent files in PREFIX
–program-prefix=PREFIX	prepend PREFIX to installed program names
–program-suffix=SUFFIX	append SUFFIX to installed program names
–enable-debug=LEVEL	enable debugging (time consuming)
–disable-mpi	disable MPI parallelization
–disable-openmp	disable OpenMP parallelization
–enable-fft3d	enable FFT3D calculation (see **[ENERGY]** section)
–with-fftw[=PATH]	use FFTW library instead of embedded FFTE routine
–with-lapack	use LAPACK
–host=host	disable compiler check (**Required** in cross compiler system)
FC	Fortran compiler command
FCFLAGS	Fortran compiler flags
LAPACK_PATH	LAPACK library path (recommended when given –with-lapack)
LAPACK_LIBS	LAPACK library path (recommended when given –with-lapack)

```
# example of high performance options on non-cross-complation system
$ ./configure

# example of enable "LAPACK"
$ ./configure --with-lapack

# example of high performance options on K computer
$ ./configure --host=k

# example of high performance options with FFT3D option on K computer
$ ./configure --enable-fft3d --host=k

# example of debugging mode with higher level (time consuming)
# enable-debug=LEVEL: 1 = without optimization
#                     2 = with debug information (-g) & -DDEBUG
#                     3 = LEVEL=2 & memory check
# if LEVEL is not defined explicitly, LEVEL is 1.
$ ./configure --enable-debug=3

# example of disabled parallelization (serial)
$ ./configure --disable-mpi --disable-openmp

# example of cross-compilation options (i.e. many computer centers)
$ ./configure --host=host
```

Once `./configure` successfully finished, `Makefile` is created for your system. To compile and install **GENESIS**, type `make install` command:

```
$ make install
```

`make install` compiles all of the **GENESIS** programs (the simulators and the analysis tools). The compiled binary files are copied to `bin/` directory. If the compilations are successfully finished, the following binary files has to be in your `bin/`:

```
$ ls -lF ../bin/
total 57840
-rwxr-xr-x  1 user  staff  6271072 10 20 18:19 atdyn
-rwxr-xr-x  1 user  staff  1804980 10 20 18:24 crd_convert
-rwxr-xr-x  1 user  staff  1911360 10 20 18:24 pcrd_convert
-rwxr-xr-x  1 user  staff  6575348 10 20 18:23 prst_setup
-rwxr-xr-x  1 user  staff  1946552 10 20 18:24 remd_convert
-rwxr-xr-x  1 user  staff  1180328 10 20 18:24 rst_convert
-rwxr-xr-x  1 user  staff  7259252 10 20 18:23 spdyn
-rwxr-xr-x  1 user  staff  1472264 10 20 18:24 trj_analysis
```

2.1.4 Regression Tests

Regression tests are prepared for **ATDYN**, **SPDYN** and **prst_setup** (parallel I/O) to check if these programs work correctly.

```
# untar the package file and change the working directory
$ tar xvhz tests.tar.bz2
$ cd tests/regression_test
$ ls -lF
total 88
drwxrwxr-x 5 user staff  4096 11 29 10:29 test_spdyn
drwxrwxr-x 3 user staff  4096 11 29 10:29 test_parallel_IO
drwxrwxr-x 5 user staff  4096 11 29 10:29 test_common
drwxrwxr-x 5 user staff  4096 11 29 10:29 test_atdyn
-rwxrwxr-x 1 user staff  8655 11 29 10:29 test.py
-rwxrwxr-x 1 user staff  8514 11 29 10:29 genesis.py
-rwxrwxr-x 1 user staff 29434 11 29 10:29 charmm.py
drwxrwxr-x 5 user staff  4096 11 29 10:29 build
drwxrwxr-x 2 user staff  4096 11 29 10:32 param
```

The tests can be run by executing a Python script (`test.py`):

```
$ ./test.py *genesis_command* *[parallel_io (optional)]*
```

Here, *genesis_command* is a command line for executing **ATDYN** or **SPDYN** (default: `mpirun -np 8 atdyn`). *parallel_io* needs to be appended to check the parallel I/O facility.

```
# execute atdyn test
$ ./test.py "mpirun -np 8 /path/to/atdyn"

# execute spdyn test
$ ./test.py "mpirun -np 8 /path/to/spdyn"

# execute spdyn test on FUJITSU compiler
$ ./test.py "mpirun -np 8 /path/to/spdyn" 0.03
```

```
# execute spdyn, parallel_io test
$ ./test.py "mpirun -np 8 /path/to/spdyn" parallel_io
```

Note: The regression tests are designed to be executed with **8** MPI processes. In particular, the parallel I/O test will be failed with other MPI conditions. As for **ATDYN** and **SPDYN** tests, other MPI conditions may give invalid results.

Note: Since the optimization scheme of FUJITSU compiler is different from those of other compilers, tolerance of spdyn test should be increased in FUJITSU compiler.

Note: **prst_setup** does not work using the Fujitsu compiler. There is no problem in running **SPDYN** by reading parallel restart files generated by **prst_setup** with other compilers.

2.2 Basic Usages

GENESIS programs use a simple command line style, where a program always interprets the first argument as an input file. The input file, *control file*, describes the parameters for controlling the program.

A typical usage of **GENESIS** program is as follow:

```
# serial execution
$ *program_name* *control_file*

# parallel execution with 8 MPI processes
$ mpirun -np 8 *program_name* *control_file*
```

For example, **SPDYN** can be called like this:

```
$ mpirun -np 8 /path/to/spdyn ctrl.inp
```

GENESIS program prints a templates of the control file then executed with `-h ctrl` *template_name* option. A list of *template_name* names can obtained by running it with `-h` option only. For example, **SPDYN** prints the template names for `md` (molecular dynamics), `min` (minimization), and `remd` (replica exchange molecular dynamics):

```
$ spdyn -h

# normal usage
% ./spdyn INP

# check control parameters of md
% ./spdyn -h ctrl md

# check control parameters of remd
% ./spdyn -h ctrl remd
```

```
# check control parameters of min
% ./spdyn -h ctrl min
(skipped..)
```

For example, the template for minimization is shown by issuing the following command:

```
$ spdyn -h ctrl min
[INPUT]
topfile = sample.top                          # topology file
parfile = sample.par                          # parameter file
psffile = sample.psf                          # protein structure file
pdbfile = sample.pdb                          # PDB file

[ENERGY]
forcefield      = CHARMM
electrostatic   = PME                         # [CUTOFF,PME]
switchdist      = 10.0                        # switch distance
(skipped...)
```

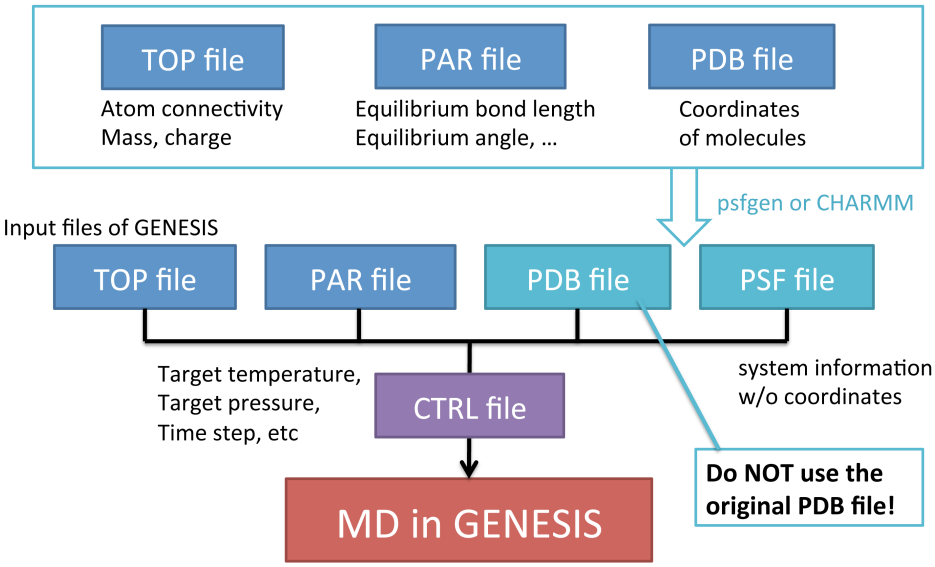
If you are interested in all of available options including detailed parameters for advanced users, add `_all` to *template_name*. For example, running with `-h ctrl min_all` prints all of available options for minimization.

2.2.1 Information Flow in GENESIS

In general, simulations of biomolecules require several types of input files in addition to the control file. For example, files for atomic coordinates, force field parameters, and molecular topology (connectivity of molecules) are necessary.

The coordinates can be given by *PDB*, *CRD*, or **GENESIS** restart files (*pdbfile*, *crdfile*, and *rstfile*, see *Input and Output files*). As the current version of **GENESIS** supports *CHARMM force fields* only, CHARMM parameter and topology files (*parfile* and *topfile*, see *Input and Output files*) are necessary along with PSF (*psffile*, see *Input and Output files*).

GENESIS does not include any programs for generating these input files. However, the setup of a simulation system could be done by using CHARMM [14] or psfgen [15] (supplied with NAMD [16] and VMD [17]). Setup procedures are demonstrated in the *Tutorial* chapters. The information flow of input files in **GENESIS** are summarized in a chart below.



2.2.2 Control File

The input files and control parameters of **GENESIS** are specified in a *control file*. The control file consists of several *sections* (i.e. **[INPUT]**, **[OUTPUT]**, **[ENERGY]**, etc.), with each section containing a set of *keywords*.

Note: Each parameter should be less than *MaxLine* (**Default : 1000**) characters. *MaxLine* is defined in "src/lib/string.fpp". If a statement is too long, it can be continued with a backslash "" at the end of the line.

Examples of the sections and their important keywords are given below:

- **[INPUT]** section: input files (see *Input and Output files*)
  - **topfile**: topology file
  - **parfile**: parameter file
  - **psffile**: protein structure file
  - **pdbfile**: PDB file
- **[OUTPUT]** section: output files (see *Input and Output files*)
  - **dcdfile**: trajectory file in DCD format
  - **rstfile**: restart file
- **[ENERGY]** section: energy and force evaluations (see *Energy*)
  - **forcefield**: type of force field (*CHARMM/KBGO*)
  - **electrostatic**: type of long-range electrostatic interactions (*CUTOFF/PME*)
  - **switchdist**: switch distance (*default: 10.0*)
  - **cutoffdist**: energy/force cutoff distance (*default: 12.0*)
  - **pairlistdist**: cutoff distance for Verlet pair-list (*default: 13.5*)

- **[DYNAMICS]** section: integrator and timestep (see *Dynamics*)
  - **integrator**: integrator (*LEAP/VVER*)
  - **nsteps**: number of MD steps (*default: 100*)
  - **timestep**: time interval of MD step (*default: 0.001*)
  - **nbupdate_period**: interval (in MD steps) of a non-bonded pair-list regeneration (*default: 10*)
- **[MINIMIZE]** section: minimization (see *Minimize*)
  - **method**: minimization algorithm (*SD*)
  - **nsteps**: number of minimization steps (*default: 100*)
  - **nbupdate_period**: interval (in minimization steps) of a non-bonded pair-list regeneration (*default: 10*)
- **[CONSTRAINTS]** section: constraints (see *Constraints*)
  - **rigid_bond**: enable constraints (*YES/NO*)
- **[ENSEMBLE]** section: temperature and pressure controls (see *Ensemble*)
  - **ensemble**: type of ensemble (*NVE / NVT / NPT*)
  - **tpcontrol**: type of thermostat and barostat (*NO / Berendsen / Langevin*)
  - **temperature**: initial and target temperature (*default: 298.15*)
  - **pressure**: target pressure (*default: 1.0*)
- **[BOUNDARY]** section: system size and boundary condition (see *Boundary*)
  - **type**: type of boundary (*default: PBC*)

2.2.3 Minimization

A control file for minimization of a molecule is shown below. **[INPUT]** section contains the input file names, and the parameters of **[ENERGY]** specify energy and force evaluation. **[MINIMIZE]** section enables the minimization algorithm.

```
[INPUT]
topfile = top_all27_prot_lipid.top            # topology file
parfile = top_all27_prot_lipid.par            # parameter file
psffile = mol.psf                             # protein structure file
pdbfile = mol.pdb                             # PDB file

[OUTPUT]
dcdfile = min.dcd                             # DCD trajectory file
rstfile = min.rst                             # GENESIS restart file

[ENERGY]
forcefield      = CHARMM                      # [CHARMM,KBGO]
electrostatic   = PME                         # [CUTOFF,PME]
switchdist      = 10.0                        # switch distance
cutoffdist      = 12.0                        # cutoff distance
pairlistdist    = 13.0                        # pairlist distance
```

```
[MINIMIZE]
nsteps          = 100                   # number of steps
eneout_period   = 10                    # energy output period
crdout_period   = 10                    # coordinates output period
rstout_period   = 100                   # restart output period

[BOUNDARY]
type            = PBC                   # [NOBC,PBC]
```

2.2.4 Molecular Dynamics

A control file for molecular dynamics of a molecule is shown below. In this case, instead of **[MINIMIZE]** section, **[DYNAMICS]** section enables the molecular dynamics algorithms. **ATDYN** and **SPDYN** give identical results for the same control file except a few cases (see the next sections for details). For small simulation systems (total number of atoms including solvent is less than 1000), **ATDYN** is recommended. In contrary, for large systems, **SPDYN** is faster than **ATDYN** as long as your machine has many cores or multiple nodes.

```
[INPUT]
topfile = top_all27_prot_lipid.top      # topology file
parfile = top_all27_prot_lipid.par      # parameter file
psffile = mol.psf                       # protein structure file
pdbfile = mol.pdb                       # PDB file
rstfile = min.rst                       # restart file

[OUTPUT]
dcdfile = md.dcd                        # DCD trajectory file
rstfile = md.rst                        # restart file

[ENERGY]
forcefield      = CHARMM                # [CHARMM,KBGO]
electrostatic   = PME                   # [CUTOFF,PME]
switchdist      = 8.0                   # switch distance
cutoffdist      = 12.0                  # cutoff distance
pairlistdist    = 13.0                  # pair-list cutoff distance

[DYNAMICS]
integrator      = LEAP                  # [LEAP,VVER]
nsteps          = 1000                  # number of MD steps
timestep        = 0.002                 # timestep (ps)
eneout_period   = 10                    # energy output period
crdout_period   = 10                    # coordinates output period
rstout_period   = 1000                  # restart output period

[CONSTRAINTS]
rigid_bond      = YES                   # constraints all bonds
                                        # involving hydrogen

[ENSEMBLE]
ensemble        = NVE                   # [NVE,NVT,NPT]
tpcontrol       = NO                    # no thermostat
temperature     = 300                   # initial temperature

[BOUNDARY]
```

```
type             = PBC                              # [NOBC,PBC]
```

CHAPTER

THREE

AVAILABLE PROGRAMS

GENESIS consists of two simulators with different algorithms and several analysis and conversion tools for trajectories.

3.1 Simulators

GENESIS has two simulators with different decomposition schemes, but Input/Output files adhering to the common formats (except for the *parallel I/O* scheme). However, some of simulation options are **ATDYN** or **SPDYN** specific.

ATDYN (ATomic decomposition DYNamics simulator)

> Molecular dynamics, energy minimization, and replica exchange molecular dynamics (REMD) [18] [19] are available. The simulator uses the atomic decomposition scheme with hybrid (MPI/OpenMP) parallelization. The simulator is designed for an easy prototyping and implementation of new methods (force fields, generalized-ensemble, external force, coarse-grained models, etc.).

SPDYN (SPatial decomposition DYNamics simulator)

> Molecular dynamics, energy minimization, and replica exchange molecular dynamics (REMD) are available. The simulator uses the spatial decomposition scheme with our new algorithms for parallel scaling; and parallel I/O scheme is available. The simulator is more complex and designed for high performance and good scalability on massively parallel computers.

3.2 Analysis tools

In addition **GENESIS** includes several tools to analyze trajectories, and to convert trajectories/restart files to intended formats. The usage of the tools is introduced in the *tutorial* chapters. (*Tutorial 1: Building and Simulating BPTI in Water*, *Tutorial 2: Simulating with Go-model*.)

trj_analysis

> An utility to analyze trajectory files generated by the simulators, by computing the values of distances/angles/dihedral angles. The utility is capable to aggregate results from separate trajectories.

crd_convert

An utility to convert trajectory files to PDB/DCD formats. In addition, it performs atomic coordinate superimposition with several algorithms, and can extract coordinates of selected atoms.

remd_convert

This utility has similar functions to **crd_convert** with an additional capability to handle REMD trajectory files. It regenerates REMD trajectory files for each conditions by reading the original trajectory files from all replicas.

3.3 Parallel I/O tools

In order to perform massive parallelization, **SPDYN** uses parallel I/O scheme, where each compute node reads/writes trajectories (coordinates, velocities, etc.) independently. The following tools are used to handle files from parallel I/O simulations. The usage of these tools is introduced in *tutorial* (chapter 16).

prst_setup

This utility provides input files for the *parallel I/O* simulation by processing huge input files (PSF, PDB) and dividing them into multiple **GENESIS** restart files. The control file is similar to that for **SPDYN**. The usage is shown in `tutorial` (in chapter 16).

pcrd_convert

This utility has similar functions to **crd_convert**. It is intended to handle *parallel I/O* trajectory files.

CHAPTER

FOUR

INPUT AND OUTPUT FILES

GENESIS uses the following input/output files. The input and output files are set in **[INPUT]** and **[OUTPUT]** sections, respectively.

4.1 Input files

GENESIS requires *topology*, *parameter*, and *psf* files describing simulation system and potential energy function. Atomic coordinates are read by either in PDB or CRD format files. If you set *rstfile*, the coordinates, velocities and other properties of the system are read from the restart file only.

---

topfile

Topology file which contains information about atom connectivity in residues and monomer unit. **GENESIS** reads *topfile* in CHARMM format. For details on the format, see the CHARMM web site [14]. **GENESIS** can read multiple topfiles (see Note). (**Required**)

parfile

Parameter file which contains force field parameters, e.g., force constants and equilibrium geometries. **GENESIS** reads *parfile* in CHARMM format. **GENESIS** can read multiple parfiles (see Note). (**Required**)

strfile

CHARMM stream file which contains additional force field parameters. **GENESIS** can read multiple strfiles (see Note). (**Optional**)

psffile

PSFfile contains system information about atomic mass, charge, atom connectivity, and etc. **GENESIS** reads *psffile* in X-PLOR and CHARMM formats. (**Required**)

pdbfile

Atomic coordinates in PDB (Protein Data Bank) format. (**Required**)

crdfile

Atomic coordinates in CHARMM format. If *crdfile* is specified, coordinates in *crdfile* are used as the initial coordinates prior to those in *pdbfile*. (**Optional**)

rstfile

This file contains atomic coordinates, velocities, simulation box size and other simulation variables with the double-precision floating-point number representation. Rstfile, which is given in **GENESIS**-original format, is specified for restarting a simulation. If *rstfile* exists, coordinates in *pdbfile*/*crdfile* and simulation box size in the control file are ignored. (**Optional**)

reffile

Reference coordinates (PDB file format) for positional restraints. This file should contain the same total number of atoms in *pdbfile*. (**Optional**)

localresfile (available for **SPDYN** only)

LocalRes file is used to applying external forces as `Local restraint`. (**Optional**)

`Local restraint` are the restraining forces implemented within the spatial decomposition scheme. Their are more computationally efficient than the ordinary ones, but the atoms has to be located in the same cell.

These energy terms are harmonic potentials:

$U(r) = k\ (r - r_0)^2$ for bonds

$U(\theta) = k\ (\theta - \theta_0)^2$ for bond angles

$U(\phi) = k\ (\phi - \phi_0)^2$ for dihedral angles

Here, r, θ, and ϕ are bond distance, angle, and dihedral angles, respectively; subscript 0 denotes their reference values; and k is the force constant. The restraint energies are added to the corresponding bond or angle/dihedral angle energies in the output file.

The file contains the following information;

```
[BOND/ANGLE/DIHEDRAL] atom atom [atom [atom]] k r0
```

`atom` indices start from 1.

Example for localres file

```
BOND      139 143           2.0 10.0
ANGLE     233 231 247       3.0 10.0
DIHEDRAL   22  24  41  43   2.0 10.0
```

Note: Ordinary restraint functions are set in **[RESTRAINTS]** section (see *Restraints*). The equivalent calculations to `local restraint` can be performed with ordinary functions of **ATDYN/SPDYN** without the spatial decomposition scheme.

Note: To specify multiple files for *parfile*, *topfile*, and *strfile*, comma-separated lists are written (e.g., parfile = par_all36_prot.prm, par_all36_na.prm, par_all36_lipid.prm). If the character string becomes long, backslash can be used for line break and continuation.

4.2 Output files

GENESIS yields trajectory files (coordinates and velocities) in *DCD* format. **GENESIS** also generates restart file (*rstfile*) during simulations for restart. Output intervals of each file are set in **[DYNAMCICS]** section.

---

dcdfile

Trajectory is written in DCD format (also used by X-PLOR and CHARMM). The double-precision flouting-point number representation is used in the file (**Required** if `crdout_period > 0`)

dcdvelfile

Velocities are also written in DCD format. (**Required** if `velout_period > 0`)

rstfile

ReSTart file contains coordinate, velocities, simulation box size and other dynamic informations. An MD simulation can be restarted using a *rstfile* generated by either a previous MD or a MIN simulation; as an REMD simulation can be restarted from any (MIN/MD/REMD) *rstfile*. (**Required** if `rstout_period > 0`)

remfile (generated in REMD simulations)

The file provides parameter ID for each replica to match replica ID and parameter ID. It is used as an input file for *remd_convert* utility. (**Required** if `exchange_period > 0`)

CHAPTER

FIVE

ENERGY

In **[ENERGY]** section, there are several options to set keywords related to energy and force evaluation.

5.1 General keywords

Force field consist of a potential energy function and a set of parameter. In the case of all-atom force field with explicit water, the potential energy function:

$$
\begin{aligned}
E(r) = &\sum_{\text{bond}} K_b(b-b_0)^2 + \sum_{\text{UB}} K_{ub}(S-S_0)^2 + \sum_{\text{angle}} K_\theta(\theta-\theta_0)^2 \\
&+ \sum_{\text{dihedral}} K_\phi(1+\cos(n\phi-\delta)) + \sum_{\text{improper}} K_\phi(\phi-\phi_0)^2 \\
&+ \sum_{\text{nonbond}} \epsilon \left[\left(\frac{R_{min,ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{R_{min,ij}}{r_{ij}} \right)^{6} \right] + \sum_{\text{nonbond}} \frac{q_i q_j}{\epsilon_1 r_{ij}}
\end{aligned}
$$

where K_b, K_{ub}, K_θ, K_ϕ, and K_ϕ are force constants of bond, UB 1-3 distance, angle, dihedral angle, and improper dihedral angle potentials, respectively; b_0, S_0, θ_0, and ϕ_0 are corresponding equilibrium values; and δ is a phase shift of the dihedral angle potential. ϵ is Lennard-Jones potential well depth, $R_{min,ij}$ is a distance of the Lennard-Jones potential minimum, q_i is an atomic charge, ϵ_1 is an effective dielectric constant, and r_{ij} is a distance between two atoms. The parameters are set according to an atom type.

The form of potential energy function is force field dependent. In **GENESIS**, TIP3P explicit water [20] with CHARMM22 and CHARMM27 force fields are available (proteins: [7][8], nucleic acids: [21][22], and lipids: [23][24]). Recently developed CHARMM36 force fields [25] [26] are also available.

As for a coarse-grained simulations, a model proposed by Karanicolas and Brooks [27] [28] is supported. It is mainly based on the Go-like model [29] with some sequence-based parameters incorporated. The potential energy function of the model is:

$$
\begin{aligned}
E(r) = & \sum_{\text{bond}} K_b(b - b_0)^2 + \sum_{\text{angle}} K_\theta(\theta - \theta_0)^2 \\
& + \sum_{\text{dihedral}} K_\phi \left(1 + \cos(n\phi - \delta)\right) \\
& + \sum_{\text{nativecontacts}} \epsilon \left[13 \left(\frac{R_{min,ij}}{r_{ij}} \right)^{12} - 18 \left(\frac{R_{min,ij}}{r_{ij}} \right)^{10} + 4 \left(\frac{R_{min,ij}}{r_{ij}} \right)^{6} \right] \\
& + \sum_{\text{nonnativepairs}} \epsilon \left[\left(\frac{R_{min,ij}}{r_{ij}} \right)^{12} \right]
\end{aligned}
$$

where the last two terms are non-bonded interactions between native contact pairs and non-native contact pairs, respectively. Roughly speaking, the native contacts are defined for the pairs close to each other in a PDB structure of a target molecule. $R_{min,ij}$ is a reference distance between a pair atoms in the PDB structure.

---

forcefield *CHARMM / KBGO*

Type of force field used for energy/force evaluation (**Default : CHARMM**).

5.2 Non-bonded interaction related keywords

The non-bonded interaction is the most time consuming part in MD simulations. Computational time of the non-bonded interactions without any approximation is proportional to $O(N^2)$. To reduce computational cost, the cut-off approximation is introduced where energy/force is truncated at the given cut-off value (keyword *cutoffdist*).

Simple spherical truncation at the cut-off value leads to discontinuities of energy and forces. So it is necessary to introduce a polynomial function (so called *switching function*) that smoothly turn off the interaction from another given value (so called *switch cut-off*) for van der Waals interactions (keyword *switchdist*). In general, there are two kinds of switching functions: switching function that smoothly turns off (1) potential energy, and (2) forces. To enable the switch function that smoothly turns off forces [30], `vdw_force_swith=YES` should be set.

The spherical truncation of electrostatic energy is different from van der Waals due it's long-range nature. To truncate electrostatic energy at the cut-off value, a shift function is introduced which is enable by `Electrostatic=Cutoff` in **GENESIS**. In case of the complete electrostatic energy, the smooth particle Mesh Ewald (PME) scheme is used (it is explained in the PME-related keyword section). Please note that the type of electrostatic energy calculation is relevant to CHARMM all-atom force field only.

---

electrostatic *CUTOFF/PME*

Type of long range electrostatic energy/force evaluations (**Default : PME**)

switchdist *Real*

Switch cut-off distance (unit : Angstrom) (**Default : 10.0**)

cutoffdist *Real*

Potential energy/force cut-off distance (unit : Angstrom) (**Default : 12.0**)

pairlistdist *Real*

Cut-off distance of Verlet pair list for non-bonded energy/force evaluations [31] (unit : Angstrom) (**Default : 13.5**)

dielec_const *Real*

Dielectric constant (**Default : 1.0**)

vdw_force_switch *YES / NO*

Usage of the force switch function for van der Waals interactions (**Default : NO**)

5.3 Particle mesh Ewald related keywords

The following keywords are relevant if `electrostatic=PME`. Electrostatic energy in the smooth particle mesh Ewald (PME) scheme [32][33] is:

$$E_{elec} = \sum_{i<j} \frac{q_i q_j}{\epsilon_1} \frac{\mathrm{erfc}(\alpha r_{ij})}{r_{ij}} + \frac{2\pi}{V} \sum_{|\mathbf{G}|^2 \neq 0} \frac{\exp(-|\mathbf{G}|^2/4\alpha^2)}{|\mathbf{G}|^2} \sum_{ij} \frac{q_i q_j}{\epsilon_1} \cos(\mathbf{G} \cdot \mathbf{r}_{ij}) - \sum_{ij} \frac{q_i q_j}{\epsilon_1} \frac{\alpha}{\sqrt{\pi}}$$

Here, the first term is calculated with cut-off because it is decreased rapidly then pair distances increase. The third term is so called *self-energy*, calculated only once. The second term is rewritten as:

$$\sum_{|\mathbf{G}|^2 \neq 0} \frac{\exp(-|\mathbf{G}|^2/4\alpha^2)}{|\mathbf{G}|^2} |\mathbf{S}(\mathbf{G})|^2$$

where

$$\mathbf{S}(\mathbf{G}) = \sum_i q_i \exp(i\mathbf{G} \cdot \mathbf{r}_i) \approx b_1(G_1) b_2(G_2) b_3(G_3) \mathbf{F}(\mathbf{Q})(G_1, G_2, G_3)$$

For the evaluation of $\mathbf{S}(\mathbf{G})$; $b_1(G_1)$, $b_2(G_2)$, and $b_3(G_3)$ are approximated by the cardinal B-splines of order n; and $\mathbf{F}(\mathbf{Q})$ is calculated by the fast Fourier transformation (FFT) of charge values.

---

pme_alpha *Real*

Exponent of complementary error function (**Default : 0.34**)

pme_ngrid_x *Integer*

Number of FFT grid points along x dimension (**Required if PME is used**)

pme_ngrid_y *Integer*

Number of FFT grid points along y dimension (**Required if PME is used**)

pme_ngrid_z *Integer*

Number of FFT grid points along z dimension (**Required if PME is used**)

pme_nspline *Integer*

B-spline order of charges for the evaluation from $b_1(G_1)$ to $b_3(G_3)$ (**Default : 4**)

pme_multiple *YES/NO*

Enable partitioning of processors for real and reciprocal PME term computation (available in **ATDYN** only) (**Default : NO**)

pme_mul_ratio *Integer*

Ratio of process numbers for real and reciprocal PME term computation (available in **ATDYN** only and used when ''PME_multiple=YES" only) (**Default : 1**)

5.3.1 Number of PME grid points

In **ATDYN** and **SPDYN**, the number of PME grid points should be multiples of 2, 3, and 5. Moreover, in **SPDYN**, there are several notes to define PME grid numbers according to processor numbers. In **SPDYN**, we first define domain numbers in each dimension such that product of them equals to the total number of MPI processors. Let us assume that the domain numbers in each dimension are domain_x, domain_y, and domain_z. If you execute configure with `--enable-fft3d` option, there are following restriction in deciding PME grid numbers:

1. pme_ngrid_x should be multiple of (2*domain_x)
2. pme_ngrid_y should be multiple of (domain_y*domain_z)
3. pme_ngrid_z should be multiple of (domain_y*domain_z) and (domain_x*domain_z)

If you do not add `--enable-fft3d` option, the restriction condition of the grid numbers are changed as following :

1. pme_ngrid_x should be multiple of (2*domain_x)
2. pme_ngrid_y should be multiple of (2*domain_y)
3. pme_ngrid_z should be multiple of domain_z

In the case of **SPDYN**, parallelization of FFT is written inside the program and both FFTE [34] and FFTW [35] libraries are used for the one-dimensional FFT. As for **ATDYN**, FFTE library is used for FFT only.

5.4 Lookup table related keywords

The following keywords are relevant if `electrostatic=PME`. For a linearly-interpolating lookup table (`table_order = 1`), table points are assigned at the unit interval of cut-off$^2/r^2$ and energy/gradients are evaluated as a function of $b_2(G_2)$ [11].

$$F(r^2) \approx F_{\text{tab}}(L) + t(F_{\text{tab}}(L+1) - F_{\text{tab}}(L))$$

where

$$L = \text{INT}(\text{Density} \times r_v^2/r^2)$$

and

$$t = \text{Density} \times r_v^2/r^2 - L$$

Density is the number of points per a unit interval. Lookup table using cubic interpolation is different from that of linear interpolation. In the case of cubic interpolation, monotonic cubic Hermite polynomial interpolation is used to impose the monotonicity of the energy value. Energy/gradients are evaluated as a function of r^2 [36] using four basis functions for the cubic Hermite spline : $h_{00}(t)$, $h_{10}(t)$, $h_{01}(t)$, $h_{11}(t)$

$$\begin{aligned} F(r^2) \approx F_{\text{tab}}(L-1)h_{00}(t) + \frac{F_{\text{tab}}(L-2) + F_{\text{tab}}(L-1)}{2}h_{10} \\ + F_{\text{tab}}(L)h_{10}(t) + \frac{F_{\text{tab}}(L-1) + F_{\text{tab}}(L)}{2}h_{11}(t) \end{aligned}$$

where

$$L = \text{INT}(\text{Density} \times r^2)$$

and

$$t = \text{Density} \times r^2 - L$$

---

table *YES / NO*

Enable the lookup table (**Default : YES**)

table_density *Integer*

Number of table points per unit interval (**Default : 20**)

table_order *0 / 1 / 3*

Order of interpolation used in the lookup table (**Default : 1 (Electrostatic=PME), 3 (Electrostatic=Cutoff**)

water_model *TIP3 / NONE*

Type of water model used for the lookup table approach (**Default : TIP3**)

CHAPTER

SIX

DYNAMICS

In **[DYNAMICS]** section, we choose integrator and time step options for MD simulations.

6.1 General keywords

In **GENESIS**,two integrators are available:velocity Verlet and leapfrog. Velocity Verlet (VVER) has two stages to update velocities $\mathbf{v}(t)$ and coordinates $\mathbf{r}(t)$ at time t with time step Δt. The update procedure is as follow:

Stage 1 :

$$\mathbf{v}(t+\frac{1}{2}\Delta t) = \mathbf{v}(t) + \frac{\Delta t}{2m}\mathbf{F}(t)$$
$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \Delta t\mathbf{v}(t+\frac{1}{2}\Delta t)$$

where m is the mass.

Force calculation $\mathbf{F}(t+\Delta t)$

Stage 2 :

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \frac{\Delta t}{2m}\left[\mathbf{F}(t) + \mathbf{F}(t+\Delta t)\right]$$

Leapfrog (LEAP) integrator has one stage. After force evaluation, velocities at half time step and coordinates at next integer time step are updated as follow:

$$\mathbf{v}(t+\frac{1}{2}\Delta t) = \mathbf{v}(t-\frac{1}{2}\Delta t) + \frac{\Delta t}{m}\mathbf{F}(t)$$
$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \Delta t\mathbf{v}(t+\frac{1}{2}\Delta t)$$

integrator *LEAP / VVER*

Type of integrator (**Default : LEAP**).

nsteps *Integer*

Number of MD steps (**Default : 100**)

timestep *Real*

Time step (unit : ps) (**Default : 0.001**)

eneout_period *Integer*

Period of energy outputs in time steps (**Default : 10**)

crdout_period *Integer*

Period of coordinates outputs in time steps (**Default : N/A**)

velout_period *Integer*

Period of velocity outputs in time steps (**Default : N/A**)

rstout_period *Integer*

Period of restart files updates in time steps (**Default : N/A**)

stoptr_period *Integer*

Period of translation and rotation motion removal in time steps (available in **ATDYN** only) (**Default : 0** (i.e. No translational rotational motion is removed.))

nbupdate_period *Integer*

Period of updating the non-bonded pair list in time steps (**Default : 10**)

pairalloc_perid *Integer*

Period of memory reallocation for Verlet pair list in time steps(available in **ATDYN** only) (**Default : 0**)

iseed *Integer*

Seed of pseudo-random number generator (if not set, it is decided from current date and time) (**Optional**)

6.2 Simulated annealing related keywords

The following keywords are relevant for MD simulations only.

---

annelaing *YES/NO*

Enable simulated annealing (**Default : NO**)

annelaing_period *Integer*

Period of temperature steps in time steps (**Default : N/A**)

dtemperature *Real*

Temperature change per temperature step (unit : Kelvin) (**Default : 0.0**)

CHAPTER

SEVEN

MINIMIZE

In **[MINIMIZE]** section, we choose options for energy minimization. Currently, the steepest descent algorithm is available only.

---

method *SD*

Algorithm of minimization (**Default : SD**).

nsteps *Integer*

Number of minimization steps (**Default : 100**)

eneout_period *Integer*

Period of energy outputs in minimization steps (**Default : 10**)

crdout_period *Integer*

Period of coordinates outputs in minimization steps (**Default : N/A**)

rstout_period *Integer*

Period of restart file updates in minimization steps (**Default : N/A**)

nbupdate_period *Integer*

Period of non-bonded pair list updates in minimization steps (**Default : 10**)

Note: If the initial structure is very deviated from the equilibrium and there are several non-bonded interactions at short distances, it is recommend to assign `table_order=0` to avoid large energy/force values. Please note, constraints are not available in minimization.

CHAPTER

EIGHT

CONSTRAINTS

In **[CONSTRAINTS]** section, constraint keywords are described. SHAKE scheme is used for bonds between heavy and hydrogen atoms [37]. For velocity Verlet integrator, RATTLE constraint is also available [38]. Bond constraints between heavy atoms are not imposed. If explict water molecules (TIP3P) are present, SETTLE algorithm is enabled automatically [39].

---

rigid_bond *YES/NO*

Enable constraints (not availalbe for minimization) (**Default : NO**).

shake_iteration *Integer*

Number of iterations for coordinates/velocities updates. If it does not converge within the given number of iterations, the program terminates with an error message. (**Default : 500**)

shake_tolerance *Real*

Tolerance of SHAKE convergence (unit : Angstrom) (**Default : 1.0e-10**)

faster_water *YES/NO*

Usage of SETTLE algorithm for constraints in water molecules (**Default : YES**) (`fast_water=NO` is not available in **SPDYN**).

CHAPTER

NINE

ENSEMBLE

In **[ENSEMBLE]** section, type of ensemble, temperature and pressure control algorithm, and parameters used in the algorithms are specified.

In the Langevin thermostat algorithm (`ensemble=NVT` with `tpcontrol=LANGEVIN`), every particle is coupled with a viscous background and a stochastic heat bath [40]:

$$\frac{d\mathbf{v}(t)}{dt} = \frac{\mathbf{F}(t) + \mathbf{R}(t)}{m} - \gamma \mathbf{v}(t)$$

where γ is the thermostat friction parameter (`gamma_t` keyword) and $\mathbf{R}(t)$ is the stochastic force. In the Langevin thermostat and barostat method (`ensemble=NPT` with `tpcontrol=LANGEVIN`), the equation of motions are given by [41]:

$$\begin{aligned}
\frac{d\mathbf{r}(t)}{dt} &= \mathbf{v}(t) + v_\epsilon \mathbf{r}(t) \\
\frac{d\mathbf{v}(t)}{dt} &= \frac{\mathbf{F}(t) + \mathbf{R}(t)}{m} - [\gamma_p + (1 + \frac{3}{f})v_\epsilon]\mathbf{v}(t) \\
\frac{dv_\epsilon(t)}{dt} &= [3V(P(t) - P_0(t)) + \frac{3K}{f} - \gamma_p v_\epsilon + R_p]/p_{mass}
\end{aligned}$$

where K is the kinetic energy, γ_p is the barostat friction parameter (`gamma_p` keyword), R_p is the stochastic pressure variable.

---

ensemble *NVE / NVT / NPT / NPAT / NPgT*

Type of ensemble (**Default : NVE**).

- **NVE**: Microcanonical ensemble
- **NVT**: Canonical ensemble
- **NPT**: Isothermal–isobaric ensemble
- **NPAT**: Constant area, normal pressure, temperature
- **NPgT**: Constant surface-tension, normal pressure, temperature

temperature *Real*

Initial and target temperature (unit : Kelvin) (**Default : 298.15**)

pressure *Real*

Target pressure in NPT or target normal pressure in NPAT and NPgT (unit : atm) (**Default : 1.0**)

gamma *Real*

Target surface tension in NPgT (unit : dyn/cm) (**Default : 0.0**)

tpcontrol *NO / BERENDSEN / LANGEVIN*

Type of thermostat and barostat (**Default : NO**). Available options are listed in Table I.

- **NO**: Do not use temperature/pressure control algorithm (for NVE only)
- **BERENDSEN**: Berendsen thermostat/barostat
- **LANGEVIN**: Langevin thermostat/barostat

tau_t *Real*

Temperature coupling time in Berendsen thermostat (unit : ps) (**Default : 5.0**)

tau_p *Real*

Pressure coupling time in Berendsen barostat (unit : ps) (**Default : 5.0**)

compressibility *Real*

Compressibility parameter in Berendsen thermostat (unit : 1/atm) (**Default : 0.0000463**)

gamma_t *Real*

Friction parameter in Langevin thermostat (unit : 1/ps) (**Default : 1.0**)

gamma_p *Real*

Friction parameter in Lengevin barostat (unit : 1/ps) (**Default : 0.1**)

isotropy *ISO/ANISO/SEMI-ISO/XY-FIXED*

Isotropy of the simulation system (**Default : ISO**). This parameter specifies how X, Y, Z dimensions of the simulation box change in NPT, NPgT, NPAT ensembles.

- **ISO**: X, Y, and Z dimensions are coupled together
- **ANISO**: X, Y, and Z dimensions fluctuate independently
- **SEMI-ISO**: X, Y, and Z dimensions fluctuate, where the ratio of X and Y dimensions are kept constant, and Z dimension changes independently
- **XY-FIXED**: X and Y dimensions are fixed, while Z dimension changes (for NPAT only)

Table I. Combinations of integrators and thermostats/barostats

tpcontrol	LEAP NVT	LEAP NPT	VVER NVT	VVER NPT
Berendsen	O	O	O	X
Langevin	O	O	O	O

CHAPTER

TEN

BOUNDARY

In **[BOUNDARY]** section, boundary conditions of the system such as simulation box size are specified.

---

type *PBC/NOBC*

Type of boundary condition (**Default : PBC**).

- **PBC**: Periodic boundary condition (rectangular or cubic box)
- **NOBC**: Non boundary condition (vacuum system) (**available in ATDYN only**).

box_size_x *Real*

Box size along the x dimension (unit : Angstrom) (**Required if PBC is used**)

box_size_y *Real*

Box size along the y dimension (unit : Angstrom) (**Required if PBC is used**)

box_size_z *Real*

Box size along the z dimension (unit : Angstrom) (**Required if PBC is used**)

domain_x *Integer*

Number of domains along the x dimension (**Optional; available in SPDYN only**)

domain_y *Integer*

Number of domains along the y dimension (**Optional; available in SPDYN only**)

domain_z *Integer*

Number of domains along the z dimension (**Optional; available in SPDYN only**)

Note: If number of domains (domain_x, domain_y, and domain_z) are not specified in the control file, they are automatically determined based on the number of MPI processes. If the user wants to specify number of domains explicitly, the product of domain numbers (domain_x x domain_y x domain_z) must be equal to the total number of MPI processes.

Note: If the simulation system has a periodic boundary condition, the user must specify the box size initially. During simulations, box size is saved in a restart file. If the restart file is used as an input of the subsequent restart MD, the initial box size is replaced with it, even if another value is specified in the control file.

CHAPTER

ELEVEN

SELECTION

To perform MD simulations with restraints like umbrella sampling, the user must select atoms to be restrained. Once the group of selected atoms is defined in this section, the group index can be called from **[RESTRAINTS]** section. Nothing is selected as default.

[SELECTION] section is also included in a control file of some analysis tools, in which selected atoms are used as analysis and fitting atoms.

---

group *expression*

Select atoms by *expression* and define them as a group. Available keywords and operators in *expression* are listed in Table II. Note that mname (or moleculename, molname) in *expression* is a molecule name that is defined by `mol_name` below.

mole_name *molecule starting-residue ending-residue*

Define *molecule* by *starting-residue* and *ending-residue*. Those residues are defined by

```
[segment id]:residue number:residue name
```

Table II. Available keywords and operators in `group`.

expression	meaning	example	other available expression
an:*name*	atom name	an:CA	atomname, atom_name
ai:*number[-[number]]*	atom index	ai:1-5	atomindex, atomidx
atno:*number[-[number]]*	atom number	atno:6	atomno
rnam:*name*	residue name	rnam:GLY	residuename, resname
rno:*number[-[number]]*	residue number	rno:1-5	residueno, resno
mname:*name*	molecule name	mname:molA	moleculename, molname
segid:*ID*	segment index	segid:PROA	segmentid, sid
water	water molecule		
hydrogen	hydrogen atoms		hydrogenatom
heavy	heavy atoms		heavyatom
backbone	protein backbone		backboneatom
all	all atoms		*
and	conjunction		&
or	logical add		|
not	negation		!
()	assemble		

Note: `ai`, `atomindex`, and `atomidx` indicate the index of atom that is sequentially renumbered over all atoms in the system, while `atno` and `atomno` are the index of atom that is assigned to each atom in PDB file. Atom index in PDB file (column 2) is not always starting from 1 or numbered sequentially. If the user want to use such PDB file as an input file, although it shoule be a rare case, `atno` and `atomno` are useful to select atoms.

Example of **[SELECTION]** section

```
[SELECTION]
group1     = resno:1-60 and an:CA
group2     = (segid:PROA and not hydrogen) | backbone
mole_name1 = molA PROA:1:TYR PROA:5:MET
group3     = mname:molA and backbone
```

CHAPTER

TWELVE

RESTRAINTS

In **[RESTRAINTS]** section contains keywords to set external harmonic restraint functions. The restraint functions are applied to the selected atom groups in **[SELECTION]** section.

External harmonic restraint functions are applied to the selected atom groups in order to restrict the motions of the atoms. The potential energy of a restaint is:

$$U(x) = k\ (x - x_0)^2$$

where x is a variable (see bellow), x_0 is a reference value, and k is a force constant.

Note: In this section, ordinary restraint functions can be set. If you want to apply local restraint functions, please use *localresfile* in **[INPUT]** (see *Input and Output files*).

---

nfunctions *Integer*

Number of ordinary restraint functions. Note that number of local restraint is not included. The following parameters is set with serial number (Maximum is **nfunctions**). (**Default: 0**)

function *POSI /DIST[MASS] /ANGLE[MASS] /DIHED[MASS] /RMSD[MASS]*

Type of harmonic restraint. (**Default: N/A**)

Each keyword is used with serial number up to *numfuncs POSI*: positional restraint. The reference coordinates are set by **reffile** in **[INPUT]**. (see *Input and Output files*)

DIST[MASS]: distance restraint.

ANGLE[MASS]: angle restraint.

DIHED[MASS]: dihedral angle restraint.

RMSD[MASS] (available in **ATDYN** only) : RMSD restraint. *MASS* means mass-weighted RMSD. The calculation is done without superimposing the reference coordinates.

DIST, *ANGLE*, *DIHED* are capable to calculate distance/angle/dihedral among representative points of the groups. *MASS* indicates that the force is applied to the center of mass of a selected group. The force without *MASS* is applied to the arithmetic average of the coordinates.

constant *Real*

Force constant of a restraint function. (**Default: 0.0**)

Unit for *DIST/RMSD* is kcal/mol/Angstrom^2 and that for *ANGLE/DIHED* is kcal/mol/rad^2. The keyword is capable to accept multiple numbers. By setting two values for *DIST*, *ANGLE*, *DIHED*, and *RMSD*, a constraint with different upper/lower force constants can be set.

reference *Real*

Reference value of a restraint function. For the position restraint, the value is ignored. (**Default: 0.0**)

Unit for *DIST* is Angstrom. Note, the unit in *ANGLE/DIHED* is degree (NOT radian) even though **constant** is kcal/mol/rad^2. The keyword is capable to accept multiple numbers. By setting two values in *DIST*, *ANGLE*, *DIHED*, and *RMSD*, a constraint with different upper/lower references can be set. (i.e. you can use a flat-bottom harmonic potential.)

select_index *Integer*

Index of a atom group, where restraint forces are applied. The index is set in **[SELECTION]**. (see *Selection*) (**Default: N/A**)

Example for **[RESTRAINTS]**

```
[RESTRAINTS]
nfunctions    = 1
function1     = DIST
reference1    = 10.0
constant1     = 2.0
select_index1 = 1 2
```

CHAPTER

THIRTEEN

REMD

In **[REMD]** section, the users can specify keywords for Replica-Exchange Molecular Dynamics (REMD) simulation. REMD method is one of the enhanced conformational sampling methods used for systems with rugged free-energy landscapes. The original temperature-exchange method (T-REMD) is the most widely used in simulations of bio-molecules [18] [42]. Here, replicas (or copies) of the original system are prepared, and different temperatures are assigned to each replica. Each replica is run in a canonical (NVT) or isobaric-isothermal (NPT) ensemble, and target temperatures are exchanged between a pair of replicas during a simulation. Exchanging temperature enforces a random walk in temperature space, resulting in the simulation surmounting energy barriers and the sampling of a much wider conformational space of target molecules.

In REMD methods, the transition probability for the replica exchange process is given by the usual Metropolis criterion,

$$w(X \to X') = \min(1, \frac{P(X')}{P(X)}) = \min(1, \exp(-\Delta)).$$

In the T-REMD method, we have

$$\Delta = (\beta_m - \beta_n) \left\{ E(q^{[j]}) - E(q^{[i]}) \right\},$$

where E is the potential energy, q is the position of atoms, β is the inverse temperature defined by $\beta = 1/k_B T$, i and j are the replica indexes, and m and n are the parameter indexes. After the replica exchange, atomic momenta are rescaled as follows:

$$p^{[i]\prime} = \sqrt{\frac{T_n}{T_m}} p^{[i]}, \qquad p^{[j]\prime} = \sqrt{\frac{T_m}{T_n}} p^{[j]},$$

where T is the temperature and p is the momenta of atoms.

The transition probability should be independent of constant temperature and constant pressure algorithms, whereas the momenta-rescaling scheme depends on the algorithm used in the simulation. If thermostat and barostat momenta are included in the equations of motion, these variables should be also rescaled after replica exchange. In GENESIS, thermostat and barostat momenta are rescaled in the case

T-REMD is performed with the Langevin method in NPT, NPAT, and NPgT ensembles. For other cases, only atomic momenta are rescaled.

In GENESIS, not only Temperature REMD but also pressure REMD, surface-tension REMD, REUS (or Hamiltonian REMD), and their multi-dimensional version are available in both **ATDYN** and **SPDYN**.

REMD simulations in GENESIS require a MPI environment. At least one MPI processors must be assigned to one replica. For example, in the case that the user wants to prepare 32 replicas, (32 x n) MPI processors are needed.

---

dimension *Integer*

Number of dimensions (i.e. number of parameter types to be exchanged) (**Default: 1**).

exchange_period *Integer*

Period of replica exchange in time steps (**Default: 100**). If `exchange_period = 0` is specified, REMD simulation without parameter exchange is executed.

iseed

Random number seed in the replica exchange scheme (**Default: 3141592**).

type *TEMPERATURE / PRESSURE / GAMMA / RESTRAINT*

Type of parameter to be exchanged in a dimension (**Default: TEMPERATURE**).

- **TEMPERATURE**: Temperature REMD
- **PRESSURE**: Pressure REMD
- **GAMMA**: Surface-tension REMD
- **RESTRAINT**: REUS (or Hamiltonian REMD)

nreplica *Integer*

Number of replicas (or parameters) in a dimension (**Default: 0**).

parameters *real* / *(real, real)*

List of parameters in a dimension. Parameters are separated by white spaces, and the total number of parameters must be equal to the above `nreplica`. In REUS (`type = RESTRAINT`), parameters should be specified in parentheses, where the first and second values separated by a comma correspond to a force constant and reference value in the restraint potential function, respectively. Even if force constants and reference values are specified here, dummy values must be set in the corresponding restraint function in **[RESTRAINTS]** section.

rest_function (only available for *REUS*)

Index of restraint function, which is pointing to the restraint function defined in **[RESTRAINTS]** section (see *Restraints*).

If `rest_function` is a positinal restraint, multiple PDB files with different coordinates should be prepared by specifying `reffile = filename{}.pdb` in **[INPUT]** section. In this case, reference values in the above `parameters` are ignored but should be specified as a dummy.

cyclic_params *YES / NO*

If `cyclic_params = YES` is specified, the first and last parameters in `parameters` are considered as neighbouring parameters (**Default: NO**). Cyclic parameter is useful when REUS in dihedral angle space is carried out.

Note: When one dimensional T-REMD, P-REMD, and surface-tension REMD is carried out, parameters specified by `temperature`, `pressure`, and `gamma` in **[ENSEMBLE]** section are ignored, respectively. For example, in the case T-REMD in the NPT ensemble is carried out, target temperature in each replica is set to the `parameters` in **[REMD]** section, while the target pressure is set to `pressure` in **[ENSEMBLE]** section

---

Note: When multi-dimensional REMD is carried out, parameters are exchanged alternatively. For example, in TP-REMD (type(1) = TEMPERATURE and type(2) = PRESSURE), temperature is exchanged at the first exchange trial, and pressure is exchanged at the second trial. This is repeated during the simulations.

Note: All parameters except for `exchange_period` in [REMD] section should not be changed before and after restart run.

Example of conventional T-REMD using 4 replicas

```
[REMD]
dimension        = 1
exchange_period  = 1000
type(1)          = TEMPERATURE
nreplica(1)      = 4
parameters(1)    = 300 310 320 330
```

Example of two-dimensinal REMD (T-REMD/REUS) using 32 replicas

```
[REMD]
dimension        = 2
exchange_period  = 1000
type(1)          = TEMPERATURE
nreplica(1)      = 8
parameters(1)    = 300 310 320 330 340 350 360 370
type(2)          = RESTRAINT
nreplica(2)      = 4
parameters(2)    = (2.0,10.0) (2.0,10.5) (2.0,11.0) (2.0,11.5)
rest_function(2) = 1

[SELECTION]
group1           = ai:1
group2           = ai:2

[RESTRAINTS]
nfunctions       = 1
function1        = DIST
reference1       = 10.0    # dummy
constant1        = 2.0     # dummy
select_index1    = 1 2
```

CHAPTER

FOURTEEN

TUTORIAL 1: BUILDING AND SIMULATING BPTI IN WATER

In this chapter, a step-by-step tutorial for a molecular dynamics simulation with **GENESIS** is provided. The aim is to guide a new user through the process of building and simulating a system containing a protein (Bovine pancreatic trypsin inhibitor: BPTI) in a box with water molecules and ions.

BPTI, a 58-residue globular protein, is a popular molecule in the simulation community because of it's small size and stability. Indeed, it is the first protein for which a molecular dynamics simulation study was performed [43] and also a 1-millisecond simulation was recently achieved with a special-purpose supercomputer [44]. In this tutorial, we examine it's stability by a 1-nanosecond simulation with **GENESIS**.

The input files for this chapter are supplied in `tutorial/bpti/` of the **GENESIS** package. Before you start the tutorial, please note the following things:

- Running all of the simulations requires several hours (or a whole day, depending on a machine specifications).
- *VMD* [17] is required for building the simulation system and the visualization of results.
- *gnuplot* [45] is required for making plots.
- PATH environment variable includes a directory with **GENESIS** binary.

If you are a Mac OSX user, it is recommended to make an alias for *VMD*:

```
$ alias vmd="/Applications/VMD\ 1.9.app/Contents/MacOS/startup.command"
```

14.1 Building a simulation system

We build a simulation system containing a BPTI in a simulation box with water molecules and ions using *VMD* and it's plugins, and generate two files required for a simulation with **GENESIS**:

1. *psffile*, containing the connectivity, mass, and charges of the molecule;
2. *pdbfile*, containing the initial structure of the molecule.

For the details about the files, see *Input and Output files*.

To build a simulation system, we download a crystal (or NMR) structure of BPTI from the RCSB Protein Data Bank (PDB) [46]. You can download the PDB file via a web browser, or directly get it by using `curl` command:

```
# change to a setup directory
$ cd tutorial/bpti/1_setup/
# download the PDB file (PDB code 4PTI)
$ curl -o 4pti.pdb http://www.rcsb.org/pdb/files/4pti.pdb
# check the downloaded file
$ less 4pti.pdb
```

Let's look at the header entries of the file. It can tell a lot about this molecule, for example, the structure was determined by X-ray diffraction (described in `EXPDTA` entry in the file), there is no missing residues (no `MISSING` entries), and there are three disulfide bonds between 5-55, 14-38, 30-51 cystein residue pairs (`SSBOND` entries).

Since atom the naming conventions of PDB and CHARMM are slightly different, we need manually to edit the file to comply with the CHARMM convention with a text editor (such as *vi* or *Emacs*). We need to replace `O` and `OXT1` atoms of the C-terminal residue (ALA58) with `OT1` and `OT2`, respectively. After the edit, ALA58 should look like follow:

```
$ less 4pti.pdb
...skipped...
ATOM    449  N   ALA A  58      25.146  29.681  -6.493  1.00 46.21       N
ATOM    450  CA  ALA A  58      25.617  30.840  -7.256  1.00 45.05       C
ATOM    451  C   ALA A  58      25.248  30.735  -8.729  1.00 46.90       C
ATOM    452  OT1 ALA A  58      24.962  31.791  -9.369  1.00 39.78       O
ATOM    453  CB  ALA A  58      27.160  30.980  -7.146  1.00 50.07       C
ATOM    454  OT2 ALA A  58      24.919  29.594  -9.172  1.00 43.54       O
TER     455      ALA A  58
...skipped...
```

Since **GENESIS** does not provide any programs for building simulation systems, we use other packages such as CHARMM [14] or psfgen [15] (supplied with NAMD [16]). In this chapter, we use *VMD* [17], and *psfgen* is called via the plug-in interface within *VMD*. Other *VMD* plug-ins (*solvate* [47] and *autoionize* [48]) are also used for solvation and neutralization, respectively.

A VMD script for building the simulation system containing a BPTI molecule in a box with water molecules and ions is at `tutorial/bpti/1_setup/setup.tcl`. We can run it with VMD:

```
$ vmd -dispdev text <setup.tcl | tee run.out
```

Here, with `-dispdev text` option, we inhibits any graphical display windows from opening (we do not need any visualization for this set-up). The redirection `<setup.tcl` instructs *VMD* to process the script. The content of the script, written in Tcl languages, is shown below.

The script consists of four parts. In the first part, the crystal water molecules are removed by using the atom selection facility of *VMD*. The protein structure without crystal water molecules is written in a *pdbfile* (`4pti_protein.pdb`). In the second part, *psfgen* plugin is called. *psfgen* matches the residues in the original file with the residues defined in *topfile* (`top_all27_prot_lipid.rtf`), and then creates disulfide bonds by patching special residues for them. Also, the coordinates of missing hydrogen atoms are guessed from the topology. Then, a *psffile* (`protein.psf`) and a *pdbfile* (`protein.pdb`) of BPTI without solvent are written. In the third part, *solvate* plugin is invoked to solvate BPTI with TIP3P bulk water molecules and fills the box for simulations under the periodic boundary conditions. The resulting system is written in `solvate.psf` and `solvate.pdb`. In the fourth part, counter ions are added to neutralize the system using *autoionize* plug-in. The final results are saved as `ionize.psf` and `ionize.pdb`.

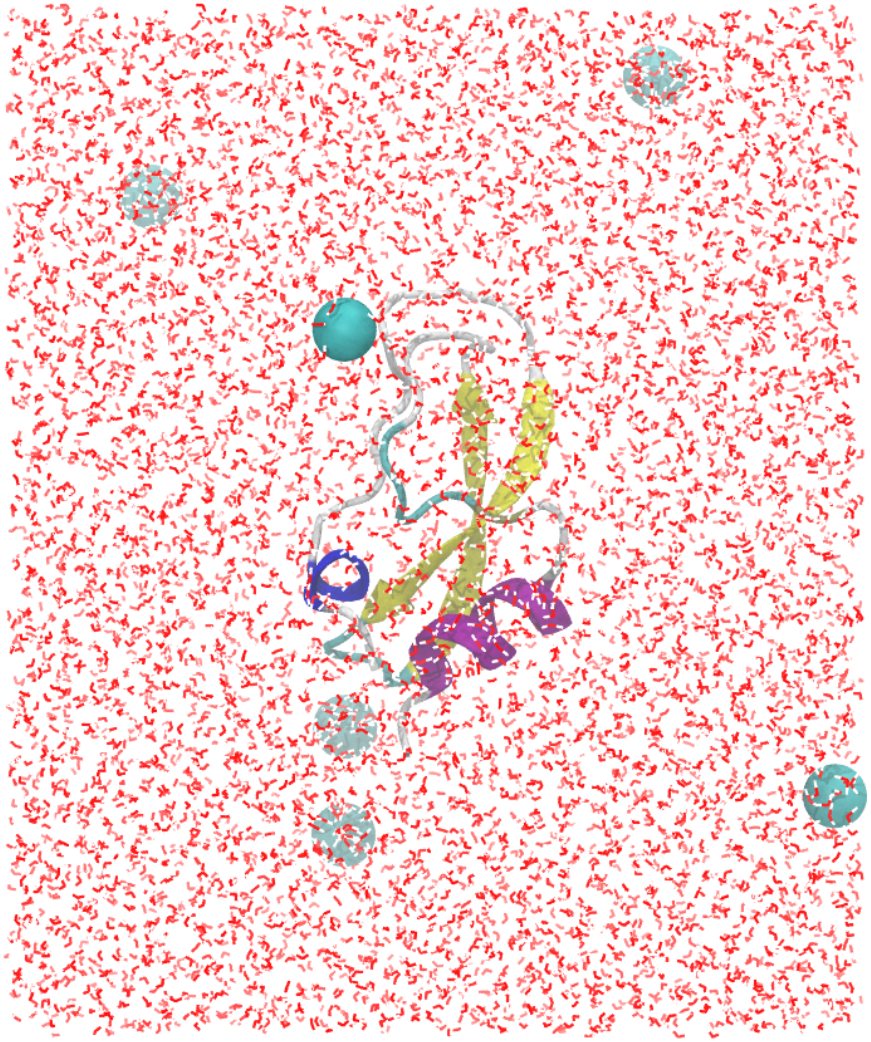
```
##### read pdb and remove crystal waters
mol load pdb 4pti.pdb

# invert the center of mass
set all [atomselect top all]
set protein [atomselect top protein]
$all moveby [vecinvert [measure center $protein weight mass]]

# write pdb of protein
$protein writepdb 4pti_protein.pdb
mol delete all

##### generate psf by using the psfgen plugin
package require psfgen
resetpsf
topology top_all27_prot_lipid.rtf

segment BPTI {
 first nter
 last cter
 pdb 4pti_protein.pdb
}
patch DISU BPTI:5 BPTI:55
patch DISU BPTI:14 BPTI:38
patch DISU BPTI:30 BPTI:51
regenerate angles dihedrals

pdbalias atom ILE CD1 CD
coordpdb 4pti_protein.pdb BPTI

guesscoord

# write psf and pdb files of protein
writepsf protein.psf
writepdb protein.pdb

#####  solvate with TIP3P waters by using the solvate plugin
mol delete all
package require solvate
solvate protein.psf protein.pdb -rotate -t 22.5 -o solvate

#####  add counter ions by using the autoionize plugin
mol delete all
package require autoionize
autoionize -psf solvate.psf -pdb solvate.pdb -neutralize -cation SOD -anion CLA -seg IO

#####  invert the center of mass and check boxsize
set all [atomselect top all]
$all moveby [vecinvert [measure center $all weight mass]]
$all writepdb ionize.pdb

set minmax [measure minmax $all]
foreach {min max} $minmax { break }
foreach {xmin ymin zmin} $min { break }
foreach {xmax ymax zmax} $max { break }
puts "Box size estimation:"
puts "boxsizex = [expr abs($xmin)+ $xmax + 1.0]"
```

```
puts "boxsizey = [expr abs($ymin)+ $ymax + 1.0]"
puts "boxsizez = [expr abs($zmin)+ $zmax + 1.0]"

exit
```

It is always a good idea to check the final structure by a visual inspection. We can visualize the structure with *VMD* by specifying *psffile* (`ionize.psf`) and *pdbfile* (`ionize.pdb`) together:

```
$ vmd -psf ionize.psf -pdb ionize.pdb
```

If the building has successfully finished, the structure looks like follows:



Note: The size (70 A x 83 A x 68 A) of the simulation box built at this step is rather large for a typical globular protein, because we are going to use **SPDYN**. For the spatial decomposition scheme in **SPDYN**, the simulation box size in each of direction has to be 5-times larger than the pair list distance (`pairlistdist=13.5` distance is in Angstrom). If you prefer a smaller simulation box, decrease the padding distance (`-t 22.5`) in *solvate* plug-in, and use **ATDYN** for your simulations instead.

14.2 Minimization with restraints on the protein

The initial structures often contain non-physical steric clashes or non-equilibrium geometries, especially at the interface between the protein and solvent, because the solvent molecules are randomly placed around the protein in the building process. Thus, before performing a molecular dynamics simulation, it is imperative to remove such clashes by minimizing the potential energy of the system. **ATDYN** and **SPDYN**, support the steepest descent algorithm for minimization, which moves atoms proportionally to their negative gradients of the potential energy.

The following command performs a 1000-step minimization with **SPDYN**:

```
# change to the minimization directory
$ cd tutorial/bpti/2_minimization/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=1
# perform minimization with SPDYN by using 8 MPI processes
$ mpirun -np 8 spdyn run.inp | tee run.out
```

The *control file* (`run.inp`, shown below) consists of several sections, such as **[INPUT]**, **[OUTPUT]**, and **[ENERGY]**, etc., where we can specify the control parameters for the calculation.

In **[INPUT]** section, input files for the minimization are specified. *topfile* (topology file), *parfile* (parameter file), *psffile* (PSF file), *pdbfile* (PDB file with an initial structure) are always required. As an optional input, *reffile* (reference file) is given as the reference structure for positional restraints (see *Input and Output files* for an explanation of each input file).

In **[OUTPUT]** section, output files are specified. **SPDYN** does not create any output file unless we explicitly specify the files. Here, *rstfile* (restart file) is used for the restart of the simulation (see *Input and Output files* for an explanation of each output file).

In **[ENERGY]** section, we set the keyword related to the energy and force evaluation. Here, the particle mesh ewald (`PME`) method is selected for computing electrostatic interactions, usually combined with the periodic boundary condition (`PBC`) in **[BOUNDARY]** section. By default, an interpolation scheme with the lookup table is used for the evaluations of non-bonded interactions.

[MINIMIZE] section turns on the minimization engine of **SPDYN**. Here, we specify 1000 steps of the steepest descent algorithm (`SD`). See *Minimize* for details.

In **[BOUNDARY]** section, the boundary condition and the simulation box size are set. Here, we use the periodic boundary condition (`PBC`). The values of the box size were taken from the previous building step. See *Boundary* for details.

In **[SELECTION]** section, we define a group (`group1`) consisting of backbone atoms of the protein to impose positional restraints. For the detailed expressions for selection, see *Selection*.

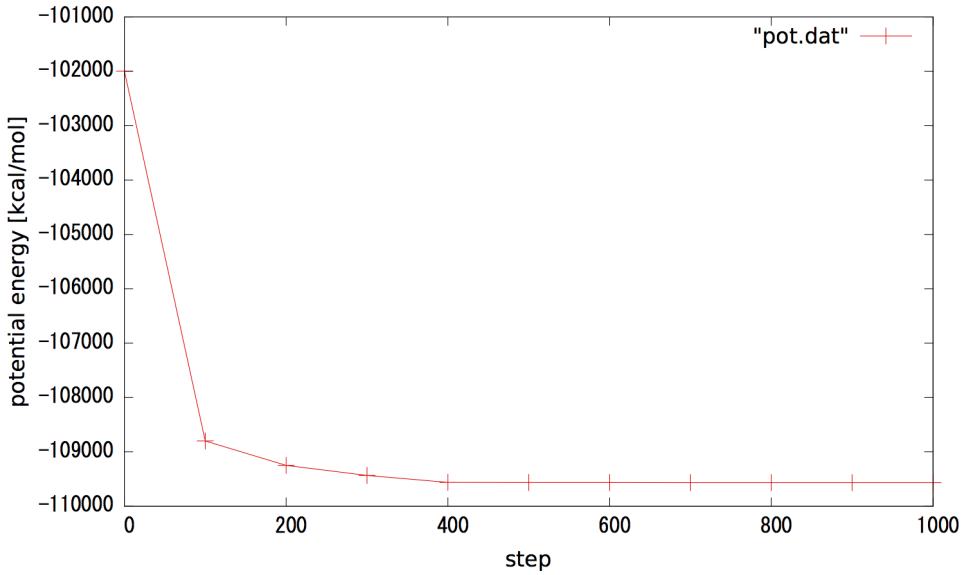
In **[RESTRAINTS]** section, positional restraints are specified for the group of backbone atoms defined in the previous **[SELECTION]** section. The positional restraints are imposed with respect to the reference coordinates given by *reffile* in **[INPUT]** section. See *Restraints* for details.

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ionize.psf               # protein structure file
pdbfile = ../1_setup/ionize.pdb               # PDB file
reffile = ../1_setup/ionize.pdb               # reference for restraints
```

```
[OUTPUT]
dcdfile = run.dcd                          # DCD trajectory file
rstfile = run.rst                          # restart file

[ENERGY]
electrostatic   =    PME                   # [CUTOFF,PME]
switchdist      =    10.0                  # switch distance
cutoffdist      =    12.0                  # cutoff distance
pairlistdist    =    13.5                  # pair-list cutoff distance
pme_ngrid_x     =    72                    # grid size_x in [PME]
pme_ngrid_y     =    80                    # grid size_y in [PME]
pme_ngrid_z     =    72                    # grid size_z in [PME]

[MINIMIZE]
nsteps          = 1000                     # number of steps
eneout_period   = 100                      # energy output period
crdout_period   = 100                      # coordinates output period
rstout_period   = 1000                     # restart output period

[BOUNDARY]
type            = PBC                      # [PBC,NOBC]
box_size_x      = 70.8250                  # box size (x) in [PBC]
box_size_y      = 83.2579                  # box size (y) in [PBC]
box_size_z      = 69.0930                  # box size (z) in [PBC]

[SELECTION]
group1          = backbone                 # index of restraint group 1

[RESTRAINTS]
nfunctions      = 1                        # number of functions
function1       = POSI                     # restraint function type
constant1       = 10.0                     # force constant
select_index1   = 1                        # restraint group
```

After the minimization, it is recommended to confirm the decrease of the potential energy. The output format of **GENESIS** is simple, so we can easily extract the potential energy term with standard UNIX tools:

```
$ grep "^INFO" run.out | tail -n +2 | awk '{print $2, $5}' >pot.dat
$ gnuplot
gnuplot> set xlabel "step"
gnuplot> set ylabel "potential energy [kcal/mol]"
gnuplot> plot "pot.dat" w lp
```



14.3 Heat-up with restraints on the protein

In this step, we heat up the system from 0.1 K to 300 K during a 10-picosecond molecular dynamics simulation. During the simulation, the positional restraints are imposed again on the protein's backbone not to disrupt the structure due to abrupt increase in temperature.

The following command performs a 10-picosecond molecular dynamics simulation, gradually heating the system from 0.1 K to 300 K with **SPDYN**:

```
# change to the heating directory
$ cd tutorial/bpti/3_heating/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=1
# perform heat-up with SPDYN by using 8 MPI processes
$ mpirun -np 8 spdyn run.inp | tee run.out
```

In **[INPUT]** section of the *control file* (`run.inp`), *rstfile* is explicitly set for the restart from the previous minimization.

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ionize.psf               # protein structure file
pdbfile = ../1_setup/ionize.pdb               # PDB file
reffile = ../1_setup/ionize.pdb               # reference for restraints
rstfile = ../2_minimization/run.rst           # restart file
```

Major differences from the previous minimization are **[DYNAMICS]**, **[CONSTRAINTS]**, and **[ENSEMBLE]** sections with the parameters for molecular dynamics simulations.

[DYNAMICS] section turns on the molecular dynamics engine of **SPDYN**. In this section, the parameters related to molecular dynamics are specified. For the heating up of the system, a simulated annealing protocol is enabled with `annealing=YES`. The simulated annealing algorithm increases the target temperature by `dtemperature` K every `anneal_period` steps. In this case, the temperature is

increased by `dtemperature=3` K every `anneal_period=50` steps. Thus, `nsteps=5000` MD steps results into temperature increase by `3*5000/50=300` K. See *Dynamics* for details.

[CONSTRAINTS] section is for constraints during a simulation. `rigid_bond` is a keyword for the SHAKE algorithm. For TIP3P waters, a faster algorithm (SETTLE) is automatically applied if `rigid_bond=YES`. See *Constraints* for details.

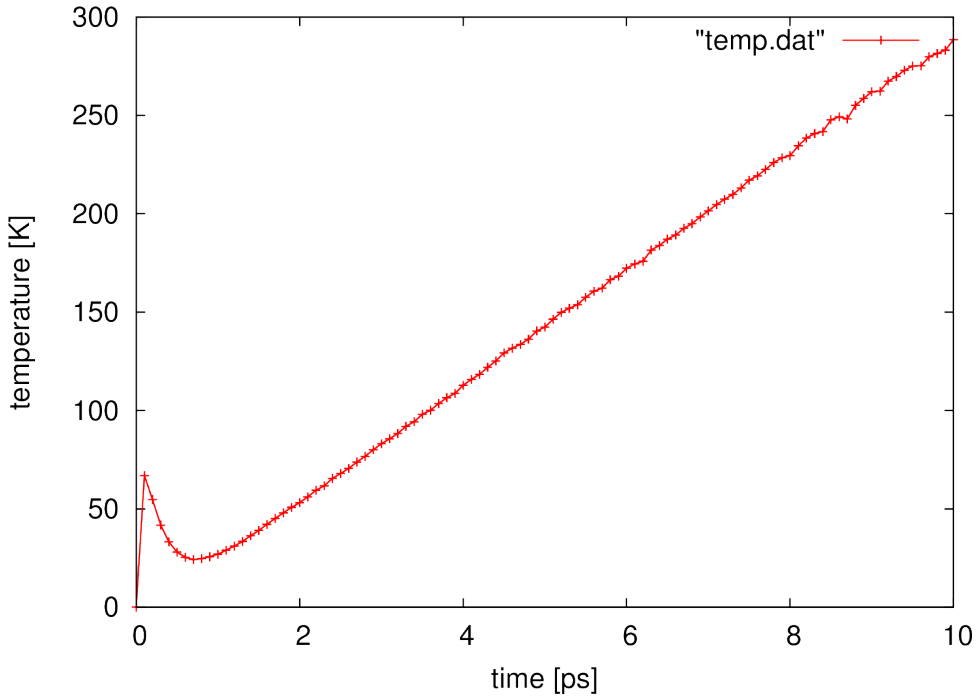
In **[ENSEMBLE]** section, we can choose a thermostat and a barostat from several standard options. For typical simulations, `LANGEVIN` or `BERENDSEN` is recommended. See *Ensemble* for details.

In **[BOUNDARY]** section, we don not set the simulation box size, since the box size values are read from the restart file.

```
[DYNAMICS]
integrator      = LEAP                          # [LEAP,VVER]
nsteps          = 5000                          # number of MD steps
timestep        = 0.002                         # time step (ps)
eneout_period   = 50                            # energy output period
crdout_period   = 50                            # coordinates output period
rstout_period   = 5000                          # restart output period
annealing       = YES                           # simulated annealing
anneal_period   = 50                            # annealing period
dtemperature    = 3                             # temperature change at
                                                # annealing (K)
[CONSTRAINTS]
rigid_bond      = YES                           # constraints of all bonds
                                                # involving hydrogen

[ENSEMBLE]
ensemble        = NVT                           # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                      # thermostat
temperature     = 0.1                           # initial temperature (K)

[BOUNDARY]
type            = PBC                           # [PBC,NOBC]
```

After the heating up simulation, the increase in temperature can be confirmed in this way:

```
$ grep "^INFO" run.out | tail -n +2 | awk '{print $3, $21}' >temp.dat
$ gnuplot
gnuplot> set xlabel "time [ps]"
gnuplot> set ylabel "temperature [K]"
gnuplot> plot "temp.dat" w lp
```



Also, it is recommended to visualize the trajectory to confirm that the protein structure is not disrupted. We can visualize the trajectory with *VMD* by the following command:

```
$ vmd -psf ../1_setup/ionize.psf -dcd run.dcd
```

Note: For real applications, 10 ps may be too short for the heating up process. For your research, longer runs than 1-nanosecond would be recommended. The same is true for the next equilibration step.

14.4 Equilibration simulation

In this step, we equilibrate the system under the condition of 300 K and 1 atm by a 10-picosecond molecular dynamics simulation. Note that the ensemble is changed from NVT to NPT. During the simulation, we no loner impose any restraints on the protein, and we relax the whole system for the next production simulation.

The following command performs a 10-picosecond molecular dynamics simulation under the condition of 300 K and 1 atm with **SPDYN**:

```
# change to the equilibration directory
$ cd tutorial/bpti/4_equilibration/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=1
# perform equilibration with SPDYN by using 8 MPI processes
$ mpirun -np 8 spdyn run.inp | tee run.out
```

In the *control file*, **[SELECTION]** and **[RESTRAINTS]** sections are removed, since we performs a restraint-free simulation. Most of the control parameters are same as those in the previous heat-up simulation, except for *rstfile*, as we specify the output of the previous simulation.

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ionize.psf               # protein structure file
pdbfile = ../1_setup/ionize.pdb               # PDB file
reffile = ../1_setup/ionize.pdb               # reference for restraints
rstfile = ../3_heating/run.rst                # restart file
```

After the equilibration, it is always a good practice to check the convergence of quantities related to thermodynamic conditions (such as temperature, pressure, or volume). This can be done in a similar way described before.

14.5 Production simulation

Finally, we perform a production 1-nanosecond molecular dynamics simulation. Running such a relatively long simulation, we can analyze the stability of BPTI in solution.

Note: Depending on the machine specifications and the number of MPI processes used, this step could take anywhere from a couple of hours to days. We can always check the progress by inspecting the standard output of **SPDYN**.

The following commands perform a 1-nanosecond molecular dynamics simulation at 300 K and 1 atm with **SPDYN**:

```
# change to the production directory
$ cd tutorial/bpti/5_production/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=1
# perform production with SPDYN by using 8 MPI processes
$ mpirun -np 8 spdyn run.inp | tee run.out
```

In **[DYNAMICS]** section of the *control file*, the total number of simulation steps is set as `nsteps=500000`, and the time step is set as `timestep=0.002` ps. This combination results in the length of `nsteps*time step` = 1000 ps = 1 ns simulation. Since the output period for coordinates is `crdout_period=500`, we have the total of `nsteps/crdout_period` = 1000 snapshots in the output trajectory.

```
[DYNAMICS]
integrator      = LEAP                      # [LEAP,VVER]
nsteps          = 500000                    # number of MD steps
timestep        = 0.002                     # time step (ps)
eneout_period   = 500                       # energy output period
crdout_period   = 500                       # coordinates output period
rstout_period   = 500000                    # restart output period
```

When the production simulation finishes, let's visualize the trajectory with *VMD*. From the visualization, you may get some insights into the fluctuations of the protein structure. In the next step, we will quantify the size of the fluctuations by analyzing the trajectory.

```
$ vmd -psf ../1_setup/ionize.psf -dcd run.dcd
```

14.6 Analysis: RMSD calculation

In this step, we examine the stability of BPTI by analyzing the trajectory obtained in the previous production simulation. As a measure to examine the stability, we calculate mean square displacements (RMSDs) from the reference structure. Roughly speaking, this measure monitors the size of the fluctuation from the reference structure. As the reference structure, we choose the crystal structure.

For the analysis, we use **crd_convert** program which is one of the post-processing programs in the **GENESIS** package. **crd_convert** is an utility to extract various quantities from the trajectory The following commands calculate the RMSDs from the trajectory of the 1-nanosecond simulation.

```
# change to the analysis directory
$ cd tutorial/bpti/6_analysis/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=1
# perform analysis with crd_convert
$ crd_convert run.inp | tee run.out
```

The *control file* for calculating the RMSDs is shown below. In **[TRAJECTORY]** section, we set the input trajectory file as `trjfile1=../5_production/run.dcd`.

Note: **crd_convert** supports multiple input files, so we can specify another file as `trjfile2=XXX`.

In **[SELECTION]** section, we define a group (`group1`) of the Cα atoms of the protein, which are fitted for the RMSD calculation. In the **[FITTING]** section, the fitting method is specified. In this case, the translations `TR` and rotations `ROT` are allowed. Finally the RMSD output file (`rmsfile=run.rms`) is set in **[OUTPUT]** section.

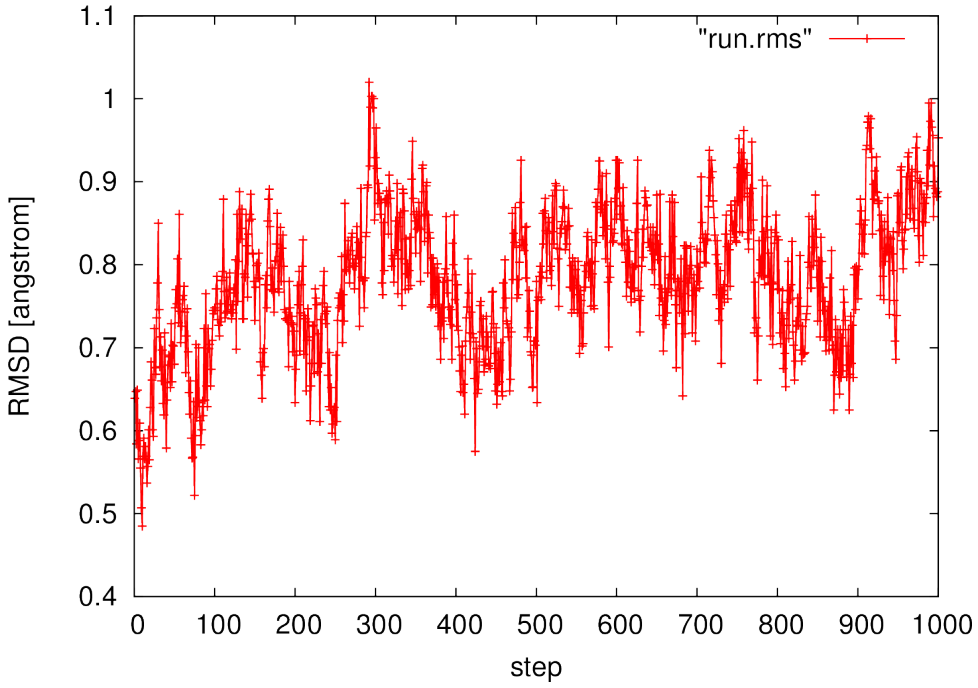
```
[INPUT]
psffile = ../1_setup/ionize.psf              # protein structure file
pdbfile = ../1_setup/ionize.pdb              # PDB file

[OUTPUT]
rmsfile = run.rms                            # RMSD file

[TRAJECTORY]
trjfile1        = ../5_production/run.dcd    # trajectory file
md_step1        = 1000                       # number of MD steps
mdout_period1   = 1                          # MD output period
ana_period1     = 1                          # analysis period
trj_format      = DCD                        # (PDB/DCD)
trj_type        = COOR+BOX                   # (COOR/COOR+BOX)

[SELECTION]
group1          = backbone and an:CA         # selection group 1

[FITTING]
fitting_method  = TR+ROT                     # method
fitting_atom    = 1                          # atom group
```

```
[OPTION]
check_only       = NO                                  # (YES/NO)
```

After running **crd_convert**, we get the RMSD file (`run.rms`). The first column of the file is for the time step, and the second column is for the RMSD values in unit of Angstrom. This can be plotted with *gnuplot*:

```
$ gnuplot
gnuplot> set xlabel "step"
gnuplot> set ylabel "RMSD [angstrom]"
gnuplot> plot "run.rms" w lp
```

The most of the RMSDs are lower than 1 Angstrom throughout the simulation, indicating that this protein is very stable in solution.



CHAPTER

FIFTEEN

TUTORIAL 2: SIMULATING WITH GO-MODEL

Coarse-grained models are useful for long-term behavior of biomolecules with time scales (typically microseconds to seconds) difficult to access by molecular dynamics simulations with atomistic models.

In this chapter, we provide a step-by-step tutorial for using a coarse-grained model (Go-model) with **GENESIS**. Using a Go-model potential energy function developed by Karanicolas and Brooks [27], we simulate folding/unfolding of protein G.

The input files of this chapter are in `tutorial/go/` of the **GENESIS** package. Before you start, please note the following things:

- The production simulation may require a whole day.
- *VMD* [17] is required for building the simulation system and the visualization of results.
- *gnuplot* [45] is required for making plots.
- PATH environment variable includes a directory with **GENESIS** binary.

If you are a Mac OSX user, it is recommended to make an alias for *VMD*:

```
$ alias vmd="/Applications/VMD\ 1.9.app/Contents/MacOS/startup.command"
```

15.1 Building a simulation system

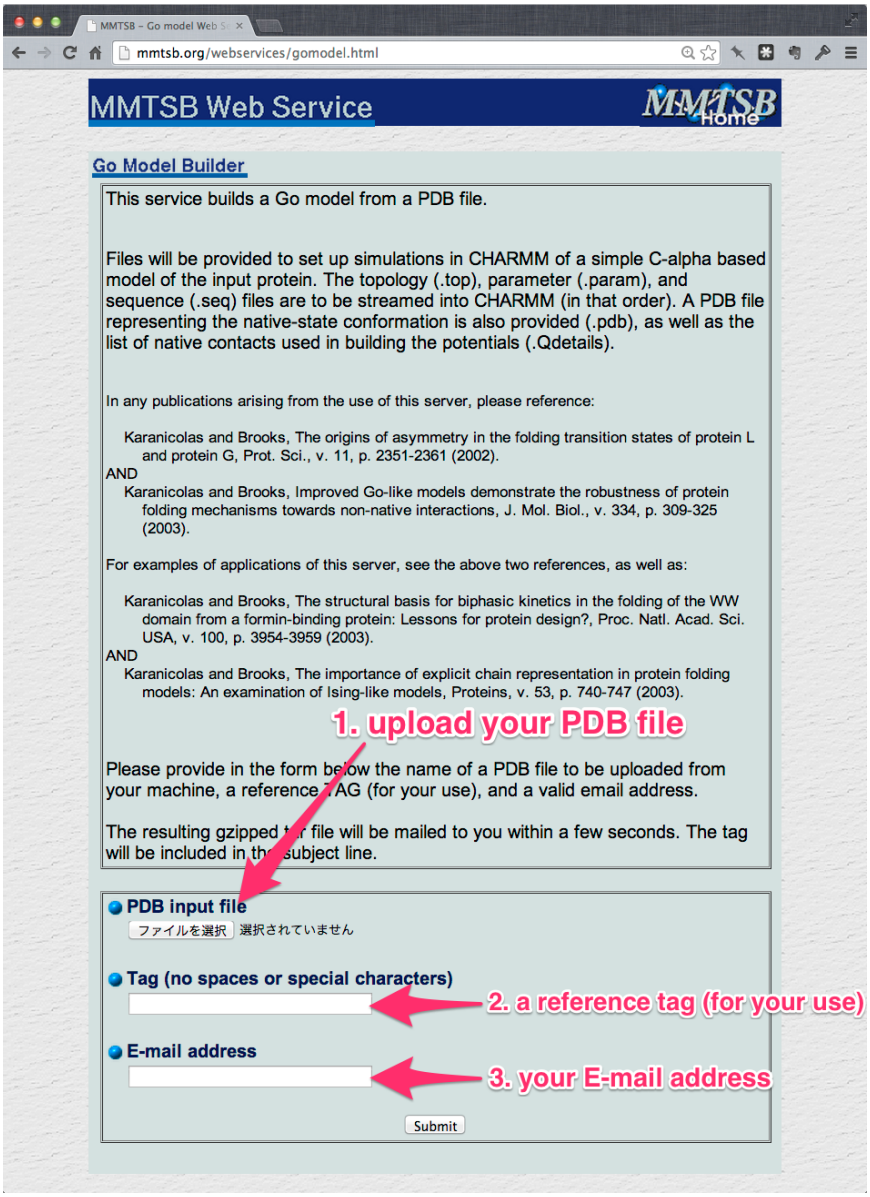
The Go-model by Karanicolas and Brooks [27] [28] is based on a minimalistic representations of a protein, where each amino acid residue is a bead located at the Cα atom position. The potential energy function is given in *Energy*. Although the function is almost same as that of CHARMM force field, there are several exceptions: (1) there are no electrostatic terms; (2) the vdW terms are divided into two terms for *native contacts* and *non-native contacts*, where the native contacts have 12-10-6 type attractive potential, while the non-native contacts have 12 type repulsive potential.

In this step, we generate *parfile* (parameter file) and *topfile* (topology file) for this Go-model using the MMTSB web service [49]. Then, from generated files, we create *psffile* (PSF file) (required for **GENESIS** simulations) using *VMD*.

To build a simulation system, we start from downloading a crystal (or NMR) structure of protein G from the RCSB Protein Data Bank (PDB) [46]. You can download the PDB file via a web browser, or directly by using `curl` command. The downloaded PDB file should be "cleaned"; the entries other than `ATOM` lines for submission to the MMTSB web service.

```
# change to the set-up directory
$ cd tutorial/go/1_setup/
# download the PDB file (PDB code 1PGB)
$ curl -o 1pgb.pdb http://www.rcsb.org/pdb/files/1pgb.pdb
# "clean" the file by deleting the entries other than ATOM
$ grep -e "^ATOM" 1pgb.pdb >1pgb_edited.pdb
```

The submission to the MMTSB web service consists of three stages: (1) upload the "cleaned" PDB file (`1pgb_edited.pdb`), (2) give a reference tag (i.e. `1pgb`), and (3) enter your e-mail address. After the submission, you'll get an e-mail with a tar-ball file containing *parfile* and *topfile*.



After extracting the tar-ball file into the working directory, you can see *parfile*, *topfile* (`GO_1pgb.param` and `GO_1pgb.top`, respectively) and other files.

```
# change to the set-up directory
$ cd tutorial/go/1_setup/
# extract the tar-ball file
```

```
$ tar xvf /path/to/GO_1pgb.tar
```

A VMD script for building the Go-model is at `tutorial/go/1_setup/setup.tcl`. We can run it with VMD:

```
$ vmd -dispdev text <setup.tcl | tee run.out
```

Here, with `-dispdev text` option, we inhibits any graphical display windows from opening (we do not need any visualization for this set-up). The redirection `<setup.tcl` instructs *VMD* to process the script. The content of the script, written in Tcl languages, is shown below.

The script consists of three parts. In the first part, the PDB file created by the MMTSB web service is read, and the molecule is moved as the center of mass is at the origin. In the second part, residue names are replaced with special names for the Go-model (G1, G2, G3,...). These replacements are required to match the residue names to those defined in *topfile* (`GO_1pgb.param`). In the third part, *psfgen* plugin is called, and *psffile* (`go.pdb`) and *pdbfile* (`go.pdb`) are generated.

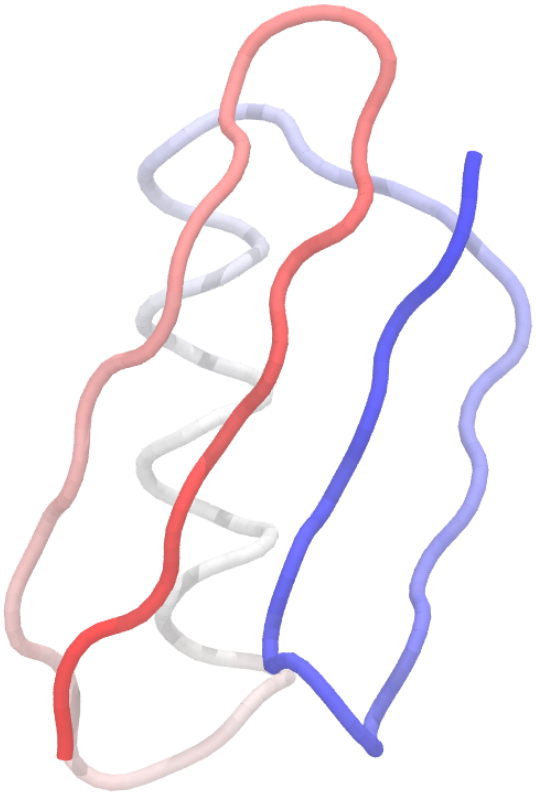
```
##### read pdb and remove center of mass
mol load pdb GO_1pgb.pdb

##### replace residue names with G1, G2, G3, ...
set all [atomselect top all]
set residue_list [lsort -unique [$all get resid]]
foreach i $residue_list {
    set resname_go [format "G%d" $i]
    set res [atomselect top "resid $i" frame all]
    $res set resname $resname_go
}

$all writepdb tmp.pdb

##### generate PSF and PDB files
package require psfgen
resetpsf
topology GO_1pgb.top

segment PROT {
 first none
 last none
 pdb tmp.pdb
}
regenerate angles dihedrals
coordpdb tmp.pdb PROT

# invert the center of mass
$all moveby [vecinvert [measure center $all weight mass]]
$all moveby [vecinvert [measure center $all]]

# write psf and pdb files
writepsf go.psf
writepdb go.pdb

exit
```

It is always a good idea to inspect the final structure visually. We can visualize the structure with *VMD*

by specifying *psffile* (`go.psf`) and *pdbfile* (`go.pdb`) together:

```
$ vmd -psf go.psf -pdb go.pdb
```

If the building has successfully finished, the structure looks like this:



15.2 Production simulation

Contrary to atomistic model simulations, coarse-grained simulations are not so sensitive to the initial structure. Thus, we do not need to relax the system (as in the case of BPTI in the previous chapter) before a production simulation. So, in this step, we are going to perform a production simulation without minimization or equilibration.

The following command performs a $5*10^8$ step production simulation with **ATDYN**:

```
# change to the production directory
$ cd tutorial/go/2_production/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=1
# perform production simulation with ATDYN by using 8 MPI processes
```

```
$ mpirun -np 8 atdyn run.inp | tee run.out
```

The *control file* (`run.inp`, shown below) is different from those for atomistic simulations.

The *control file* (`run.inp`) contains several sections, such as **[INPUT]**, **[OUTPUT]**, and **[ENERGY]**, where we can specify the control parameters for the simulation. In **[INPUT]** section, *topfile* (topology file), *parfile* (parameter file), *psffile* (PSF file), *pdbfile* (the initial structure) are set (see *Input and Output files* for an explanation of each input file).

In **[OUTPUT]** section, output files are set. **ATDYN** does not create any output file unless we explicitly specify the files. Here, *rstfile* (restart file) and *dcdfile* (binary trajectory file) are set (see *Input and Output files* for an explanation of each output file).

In **[ENERGY]** section, we can specify the parameters related to the energy and force evaluation. Here, `KBGO` is specfied for the Go-model of Karanicolas and Brooks (`forcefield=KBGO`). This value turns on the special 12-10-6 type vdW interactions for native contacts. For the distances, very large values are set (`switchdist=997`, `cutoffdist=998`, `pairlistdist=999`) to perform a "non-cutoff" simulation.

[DYNAMICS] section turns on the molecular dynamics engine of **ATDYN**. For the Go-model with SHAKE constraints, time step can be 20 fs.

In **[CONSTRAINTS]** section, we enable SHAKE algorithm on all bonded pairs (`rigid_bond=YES`). To supress SETTLE algorithm applied for non-existent TIP3P water molecules, we have to disable it explicitly (`fast_water=NO`). The tolerance for SHAKE is rather large compared to atomistic simulations (`shake_tolerance=1.0e-6`).

In **[ENSEMBLE]** section, `LANGEVIN` thermostat is chosen for an isothermal simulation with the friction constant of 1.0 ps^{-1}.

Finally, in **[ENSEMBLE]** section, no boundary condition is imposed in this system.

```
[INPUT]
topfile = ../1_setup/GO_1pgb.top              # topology file
parfile = ../1_setup/GO_1pgb.param            # parameter file
psffile = ../1_setup/go.psf                   # protein structure file
pdbfile = ../1_setup/go.pdb                   # PDB file

[OUTPUT]
dcdfile = run.dcd                             # DCD trajectory file
rstfile = run.rst                             # restart file

[ENERGY]
forcefield      = KBGO
electrostatic   = CUTOFF
switchdist      = 997.0                       # switch distance
cutoffdist      = 998.0                       # cutoff distance
pairlistdist    = 999.0                       # pair-list cutoff distance
table           = NO                          # usage of lookup table

[DYNAMICS]
integrator      = LEAP                        # [LEAP,VVER]
nsteps          = 100000000                   # number of MD steps
timestep        = 0.020                       # timestep (ps)
eneout_period   = 10000                       # energy output period
rstout_period   = 10000                       # restart output period
crdout_period   = 10000                       # coordinates output period
```

```
nbupdate_period = 10000                    # nonbond update period

[CONSTRAINTS]
rigid_bond      = YES                      # constraints all bonds
                                           # involving hydrogen
fast_water      = NO                       # settle constraint
shake_tolerance = 1.0e-6                   # tolerance (Angstrom)

[ENSEMBLE]
ensemble        = NVT                      # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                 # thermostat
temperature     = 325                      # initial and target
                                           # temperature (K)
gamma_t         = 0.01                     # thermostat friction (ps-1)
                                           # in [LANGEVIN]
[BOUNDARY]
type            = NOBC                     # [PBC, NOBC]
```

15.3 Analysis: RMSD calculation

For the analysis, we use **crd_convert** program which is one of the post-processing programs in the **GENESIS** package. **crd_convert** is a utility to derive various quantities from the trajectory. The following commands calculate the RMSDs from the trajectory of the simulation.

```
# change to the analysis directory
$ cd tutorial/go/3_analysis/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=1
# perform analysis with crd_convert
$ crd_convert run.inp | tee run.out
```

The *control file* for calculating the RMSDs is shown below. In **[TRAJECTORY]** section, we set the input trajectory file as `trjfile1=../2_production/run.dcd`.

Note: **crd_convert** supports multiple input files, so we can specify another file as `trjfile2=XXX`.

In **[SELECTION]** section, we define a group (`group1`) of all beads in the model which are fitted for the RMSD calculation. In the **[FITTING]** section, the fitting method is specified. In this case, the translations `TR` and rotations `ROT` are allowed for the fitting. Finally the RMSD output file (`rmsfile=run.rms`) is set in **[OUTPUT]** section.

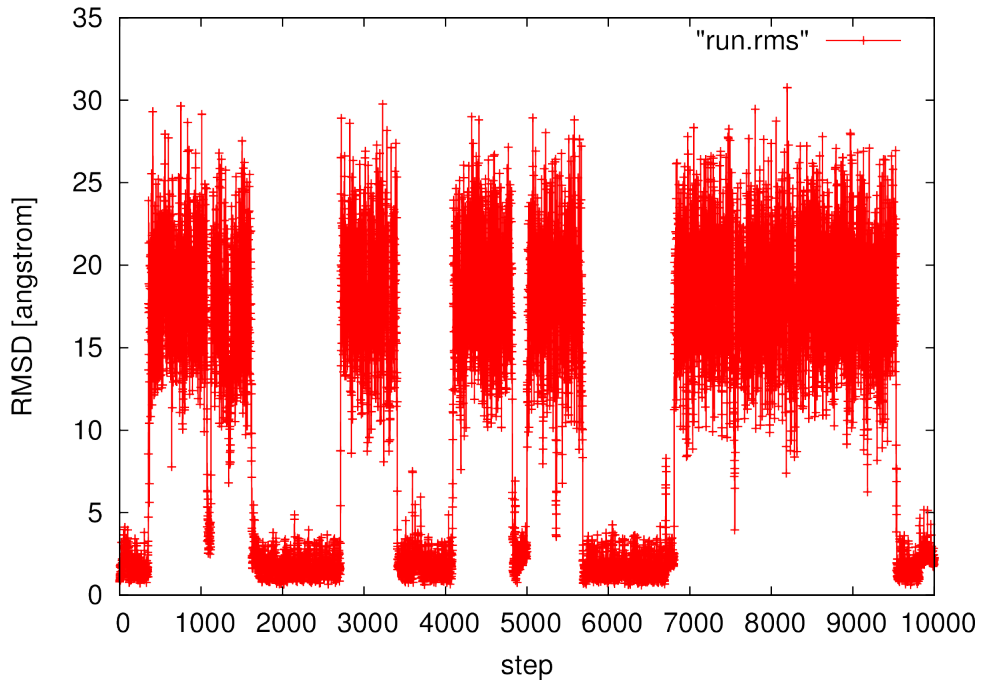
```
[INPUT]
psffile = ../1_setup/go.psf                # protein structure file
reffile = ../1_setup/go.pdb                # PDB file

[OUTPUT]
rmsfile = run.rms                          # RMSD file

[TRAJECTORY]
trjfile1         = ../2_production/run.dcd # trajectory file
md_step1         = 100000000               # number of MD steps
```

```
mdout_period1     = 10000                       # MD output period
ana_period1       = 1                           # analysis period
trj_format        = DCD                         # (PDB/DCD)
trj_type          = COOR                        # (COOR/COOR+BOX)

[SELECTION]
group1            = all                         # selection group 001

[FITTING]
fitting_method    = TR+ROT                      # method
fitting_atom      = 1                           # atom group

[OPTION]
check_only        = NO                          # (YES/NO)
```

After running **crd_convert**, we get the RMSD file (`run.rms`). The first column of the file is for the time step, and the second column is for the RMSD values in unit of Angstrom. This can be plotted with *gnuplot*:

```
$ gnuplot
gnuplot> set xlabel "step"
gnuplot> set ylabel "RMSD [angstrom]"
gnuplot> plot "run.rms" w lp
```

The noticeable fluctuations of RMSD values correspond to folding/unfolding events of the protein.



CHAPTER

SIXTEEN

TUTORIAL 3: REMD SIMULATION OF ALANINE DIPEPTIDE

In this chapter, a tutorial for a replica-exchange molecular dynamics (REMD) simulation with **GENESIS** is provided. The aim is to introduce a new **GENESIS** user how to perform REMD simulation of alanine dipeptide (ALAD) in water. ALAD has often been used as a model in theoretical studies of backbone conformational analyses. Usage of the REMD function in **GENESIS** is basically same between **ATDYN** and **SPDYN**. Hereafter, we show an example for ATDTN.

The input files of this chapter are in `tutorial/alad_remd/` of the **GENESIS** package. Before you start the tutorial, please note the following things:

- Running all of the simulations requires several days.
- *VMD* [17] is required for building a simulation system and the visualization of results.
- *gnuplot* [45] is required for making plots.
- PATH environment variable includes a directory with **GENESIS** binary.

If you are a Mac OSX user, it is recommended to make an alias for *VMD*:

```
$ alias vmd="/Applications/VMD\ 1.9.app/Contents/MacOS/startup.command"
```

`tutorial/alad_remd/1_setup` directory includes *topfile* (topology file), *parfile* (parameter file), *psffile* (PSF file), *pdbfile* (initial structure) required for the REMD simulation of ALAD.

16.1 Minimization

As the initial structures often contain non-physical steric clashes or non-equilibrium geometries, it is recommended to relax the system before the REMD simulation by minimizing the potential energy of the system. **ATDYN** and **SPDYN** support the steepest descent algorithm for minimization, which moves atoms proportionally to their negative gradients of the potential energy.

The following commands perform a 1000-step minimization with **ATDYN**.

```
# change to the minimization directory
$ cd tutorial/alad_remd/2_minimization/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=1
```

```
# perform minimization with ATDYN by using 8 MPI processes
$ mpirun -np 8 atdyn remd_min.inp | tee remd_min.out
```

The *control file* (`remd_min.inp`, shown below) consists of several sections, such as **[INPUT]**, **[OUTPUT]**, and **[ENERGY]**, etc., where we can specify the control parameters for the simulation.

In **[INPUT]** section, input files for the minimization are specified. *topfile* (topology file), *parfile* (parameter file), *psffile* (PSF file), *pdbfile* (initial structure) are always required. An optional input, *reffile* (reference file), is the reference structure for positional restraints (see *Input and Output files* for an explanation about each input file).

In **[OUTPUT]** section, output files are specified. **ATDYN** does not create any output file unless we explicitly specify the files. Here, *rstfile* (restart file) is used for the restart of the simulation (see *Input and Output files* for an explanation about each output file).

In **[ENERGY]** section, we set the keyword related to the energy and force evaluation. Here, the particle mesh ewald (`PME`) method is selected for computing electrostatic interactions, usually combined with the periodic boundary condition (`PBC`) in **[BOUNDARY]** section. `table_order=0` is specified for non-bonded interactions (by default, an interpolation scheme with the lookup table is used for the evaluations of non-bonded interactions). When `table_order=0`, the forces values of the nearest grid point in the table are used (without any interpolations), and thus large forces which may occur at the beginning of minimization can be safely truncated. See *Energy* for details.

[MINIMIZE] section turns on the minimization engine of **ATDYN**. Here, we specify 1000 steps of the steepest descent algorithm (`SD`). See *Minimize* for details.

In **[BOUNDARY]** section, the boundary condition and the simulation box size are set. Here, we use the periodic boundary condition (`PBC`). See *Boundary* for details.

In **[SELECTION]** section, we define a group (`group1`) consisting of backbone atoms of the protein to impose positional restraints. For the detailed expressions for selection, see *Selection*.

In **[RESTRAINTS]** section, positional restraints are specified for the group of backbone atoms defined in the previous **[SELECTION]** section. The positional restraints are imposed with respect to the reference coordinates given by *reffile* in **[INPUT]** section. See *Restraints* for details.

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ala.psf                  # protein structure file
pdbfile = ../1_setup/ala.pdb                  # PDB file
reffile = ../1_setup/ala.pdb                  # reference for restraints

[OUTPUT]
dcdfile = ./remd_min.dcd                      # DCD trajectory file
rstfile = ./remd_min.rst                      # restart file

[ENERGY]
electrostatic   = PME                         # [CUTOFF,PME]
switchdist      = 7.5                         # switch distance
cutoffdist      = 8.0                         # cutoff distance
pairlistdist    = 9.0                         # pair-list distance
table           = YES                         # usage of lookup table
table_order     = 1                           # order of lookup table
pme_ngrid_x     = 64                          # grid size_x in [PME]
pme_ngrid_y     = 64                          # grid size_y in [PME]
```

```
pme_ngrid_z     = 64                            # grid size_z in [PME]

[MINIMIZE]
nsteps          = 1000                          # number of steps
eneout_period   = 10                            # energy output period
crdout_period   = 10                            # coordinates output period
rstout_period   = 1000                          # restart output period
nbupdate_period = 5                             # nonbond update period

[BOUNDARY]
type            = PBC                           # [PBC,NOBC]
box_size_x      = 64.0                          # box size (x) in [PBC]
box_size_y      = 64.0                          # box size (y) in [PBC]
box_size_z      = 64.0                          # box size (z) in [PBC]

[SELECTION]
group1          = backbone                      # index of restraint group 1

[RESTRAINTS]
nfunctions      = 1                             # number of functions
function1       = POSI                          # restraint function type
constant1       = 10.0                          # force constant
select_index1   = 1                             # restraint group
```

16.2 Equilibration

In this step, we equilibrate the system under the condition of 300 K and 1 atm by a 10-picosecond molecular dynamics simulation. In the first equilibration (eq1), positional restraints are imposed on the backbone, and at the second equilibration (eq2) all the restraints are removed to relax the whole system for the production simulation.

The following commands perform the first 10-picosecond NPT molecular dynamics simulation under the condition of 300 K and 1 atm with restraints (eq1).

```
# change to the equilibration directory
$ cd tutorial/alad_remd/3_equilibration/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=1
# perform equilibration with ATDYN by using 8 MPI processes
$ mpirun -np 8 atdyn remd_eq1.inp | tee remd_eq1.out
```

In **[INPUT]** section of the *control file* (`remd_eq1.inp`), *rstfile* is set to the restart file from the previous minimization.

The main differences from the previous minimization step-up are **[DYNAMICS]**, **[CONSTRAINTS]**, and **[ENSEMBLE]** sections, which contain the keywords for molecular dynamics simulations.

[DYNAMICS] section enables the molecular dynamics engine of **ATDYN**. Parameters related to molecular dynamics integrator are specified in this section. See *Dynamics* for details.

[CONSTRAINTS] section specifies constraints during a simulation. `rigid_bond` is a keyword for the SHAKE algorithm. For TIP3P waters, a more fast algorithm (SETTLE) is automatically applied if `rigid_bond=YES`. See *Constraints* for details.

In **[ENSEMBLE]** section, we can choose a thermostat and a barostat from several standard options. For typical simulations, `LANGEVIN` or `BERENDSEN` is recommended. See *Ensemble* for details.

In **[BOUNDARY]** section, we no longer need to specify the simulation box size, as the box size are read from the restart file.

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ala.psf                  # protein structure file
pdbfile = ../1_setup/ala.pdb                  # PDB file
rstfile = ../2_minimization/remd_min.rst      # restart file
reffile = ../1_setup/ala.pdb                  # reference for restraints

[OUTPUT]
dcdfile = ./remd_eq1.dcd                      # DCD trajectory file
rstfile = ./remd_eq1.rst                      # restart file

[ENERGY]
electrostatic   = PME                         # [CUTOFF,PME]
switchdist      = 7.5                         # switch distance
cutoffdist      = 8.0                         # cutoff distance
pairlistdist    = 9.0                         # pair-list distance
table           = YES                         # usage of lookup table
table_order     = 1                           # order of lookup table
pme_ngrid_x     = 64                          # grid size_x in [PME]
pme_ngrid_y     = 64                          # grid size_y in [PME]
pme_ngrid_z     = 64                          # grid size_z in [PME]

[DYNAMICS]
integrator      = LEAP                        # [LEAP,VVER]
nsteps          = 5000                        # number of MD steps
timestep        = 0.002                       # time step (ps)
eneout_period   = 50                          # energy output period
crdout_period   = 50                          # coordinates output period
rstout_period   = 5000                        # restart output period
nbupdate_period = 5                           # nonbond update period

[CONSTRAINTS]
rigid_bond      = YES                         # constraints all bonds
                                              # involving hydrogen

[ENSEMBLE]
ensemble        = NPT                         # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                    # thermostat and barostat
temperature     = 300.0                       # initial and target
                                              # temperature (K)
pressure        = 1.0                         # target pressure (atm)

[BOUNDARY]
type            = PBC                         # [PBC,NOBC]

[SELECTION]
group1          = backbone                    # index of restraint group 1

[RESTRAINTS]
nfunctions      = 1                           # number of functions
function1       = POSI                        # restraint function type
```

```
constant1       = 5.0                           # force constant
select_index1   = 1                             # restraint group
```

In the second equilibration, **[SELECTION]** and **[RESTRAINTS]** sections in the *control file* are removed, as we perform a restraint-free simulation (eq2). Most of the control parameters are the same to the previous equilibration (eq1) except *rstfile*, which is set to the *restart file* of the first equilibration.

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ala.psf                  # protein structure file
pdbfile = ../1_setup/ala.pdb                  # PDB file
rstfile = ./remd_eq1.rst                      # restart file
reffile = ../1_setup/ala.pdb                  # reference for restraints

[OUTPUT]
dcdfile = ./remd_eq2.dcd                      # DCD trajectory file
rstfile = ./remd_eq2.rst                      # restart file

[ENERGY]
electrostatic   = PME                         # [CUTOFF,PME]
switchdist      = 7.5                         # switch distance
cutoffdist      = 8.0                         # cutoff distance
pairlistdist    = 9.0                         # pair-list distance
table           = YES                         # usage of lookup table
table_order     = 1                           # order of lookup table
pme_ngrid_x     = 64                          # grid size_x in [PME]
pme_ngrid_y     = 64                          # grid size_y in [PME]
pme_ngrid_z     = 64                          # grid size_z in [PME]

[DYNAMICS]
integrator      = LEAP                        # [LEAP,VVER]
nsteps          = 5000                        # number of MD steps
timestep        = 0.002                       # time step (ps)
eneout_period   = 50                          # energy output period
crdout_period   = 50                          # coordinates output period
rstout_period   = 5000                        # restart output period
nbupdate_period = 5                           # nonbond update period

[CONSTRAINTS]
rigid_bond      = YES                         # constraints all bonds
                                              # involving hydrogen

[ENSEMBLE]
ensemble        = NPT                         # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                    # thermostat and barostat
temperature     = 300.0                       # initial and target
                                              # temperature (K)
pressure        = 1.0                         # target pressure (atm)

[BOUNDARY]
type            = PBC                         # [PBC]
```

Also, it is recommended to visualize the trajectory to confirm, the protein structure is not disrupted. We can visualize the trajectory with *VMD* by the following command:

```
$ vmd -psf ../1_setup/ala.psf -dcd remd_eq2.dcd
```

Note: For real applications, 10 ps may be too short for equilibration. For your research, longer equilibration than 1 ns is recommended.

After the equilibration, it is always a good practice to check the convergence of quantities related to thermodynamic conditions (such as temperature, pressure, or volume). This can be done in a similar way described before.

16.3 Replica temperatures

To perform REMD simulation we need to determine the number of replicas and their temperatures. These parameters have to be chosen carefully, as their are one of the most important factors affecting the results of REMD simulation.

For this purpose we recommend to use *Temperature generator for REMD-simulations* [50]. This web server automatically generate number of replicas and their temperatures according to provided information.

16.4 Production simulation

Before performing REMD simulation, the system needs to be equilibrated at each temperature. The following commands perform a 20-picosecond NVT molecular dynamics simulations.

```
# change to the production directory
$ cd tutorial/alad_remd/4_production/
# perform equilibration at each temperature with ATDYN by
  submitting batch job script
$ qsub remd_run_eq.sh
```

The batch job script *remd_run_eq.sh* is at `tutorial/alad_remd/4_production/` for reference.

In **[OUTPUT]** section of the *control file*, *logfile* gives log file for each replica and *remfile* gives replica exchange parameter ID file. In this section "{}" returns a series of 32 output files (number of replicas).

In **[REMD]** section of the *control file*, the number of dimensions is set to `dimension=1`, and the period between replica exchanges is set to `exchange_period=0` for equilibration (no need to be exchanged in this step). Type of exchanged variable in each dimension is set to `type(1)=TEMPERATURE`, the number of replicas in each dimension is set to `nreplica(1)=32`, and the variables in each dimension are set to `parameters(1)=300 301 ... 331`. `cyclic_params(1)=NO` determines if **ATDYN** performs replica exchanges cyclically or not. See *REMD* for details.

In **[DYNAMICS]** section of the *control file*, the total number of simulation steps is set to `nsteps=10000`, and the time step is set to `timestep=0.002 ps`. This results into a `nstep*timestep = 20 ps` simulation. Since the output period for coordinates is set to `crdout_period=50`, we have the total of `nsteps/crdout_period=200` snapshots in the output trajectory.

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ala.psf                  # protein structure file
pdbfile = ../1_setup/ala.pdb                  # PDB file
rstfile = ../3_equilibration/remd_eq2.rst     # restart file

[OUTPUT]
logfile = remd_run_eq{}.log                 # log file of each replica
dcdfile = remd_run_eq{}.dcd                 # DCD trajectory file
rstfile = remd_run_eq{}.rst                 # restart file
remfile = remd_run_eq{}.rem                 # replica exchange ID file

[REMD]
dimension       = 1
exchange_period = 0

type(1)         = TEMPERATURE
nreplica(1)     = 32
parameters(1)   = 300 301 302 303 304 305 306 307 308 309 310 311 312 \
                  313 314 315 316 317 318 319 320 321 322 323 324 325 \
                  326 327 328 329 330 331
cyclic_params(1)= NO

[ENERGY]
electrostatic   = PME                         # [CUTOFF,PME]
switchdist      = 7.5                         # switch distance
cutoffdist      = 8.0                         # cutoff distance
pairlistdist    = 9.0                         # pair-list distance
table_order     = 1                           # order of lookup table
pme_ngrid_x     = 64                          # grid size_x in [PME]
pme_ngrid_y     = 64                          # grid size_y in [PME]
pme_ngrid_z     = 64                          # grid size_z in [PME]

[DYNAMICS]
integrator      = LEAP                        # [LEAP,VVER]
nsteps          = 10000                       # number of MD steps
timestep        = 0.002                       # time step (ps)
eneout_period   = 50                          # energy output period
crdout_period   = 50                          # coordinates output period
rstout_period   = 10000                       # restart output period
nbupdate_period = 5                           # nonbond update period

[CONSTRAINTS]
rigid_bond      = YES                         # constraints all bonds
                                              # involving hydrogen

[ENSEMBLE]
ensemble        = NVT                         # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                    # thermostat
temperature     = 300.0                       # initial and target
                                              # temperature (K)
[BOUNDARY]
type            = PBC                         # [NOBC,PBC]
box_size_x      = 64.0                        # box size (x) in [PBC]
box_size_y      = 64.0                        # box size (y) in [PBC]
```

```
box_size_z      = 64.0                          # box size (z) in [PBC]
```

After the equilibrating the system, we move on to the production simulation of REMD. The following commands perform 1-nanosecond REMD simulations in NVT ensemble:

```
# perform production simulation of REMD with ATDYN by submitting a
batch job script
$ qsub remd_run.sh
```

The batch job script *remd_run.sh* is at `tutorial/alad_remd/4_production/` for reference.

Differences from the previous equilibration step are **[INPUT]**, **[REMD]** and **[DYNAMICS]** sections.

In **[INPUT]** section of the *control file* (`remd_run.inp`), *rstfile* is set to the restart from the equilibration. In this section "{}" also returns a series of 32 input files.

In **[REMD]** section of the *control file*, the period between replica exchanges is set to `exchange_period=1000`. See *REMD* for details. You can break a long line into multiline using backslash.

In **[DYNAMICS]** section of the *control file*, the total number of simulation steps is set to''nsteps=1500000'', and the time step is set to `timestep=0.002 ps`. This results into a `nstep*timestep = 3 ns` simulation. Since the output period for coordinates is set to `crdout_period=1000`, we have the total of `nsteps/crdout_period=1500` snapshots in the output trajectory.

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ala.psf                  # protein structure file
pdbfile = ../1_setup/ala.pdb                  # PDB file
rstfile = ./remd_run_eq{}.rst                 # restart file

[OUTPUT]
logfile = remd_run{}.log                    # log file of each replica
dcdfile = remd_run{}.dcd                    # DCD trajectory file
rstfile = remd_run{}.rst                    # restart file
remfile = remd_run{}.rem                    # replica exchange ID file

[REMD]
dimension       = 1
exchange_period = 1000
type(1)         = TEMPERATURE
nreplica(1)     = 32
parameters(1)   = 300 301 302 303 304 305 306 307 308 309 310 311 312 \
                  313 314 315 316 317 318 319 320 321 322 323 324 325 \
                  326 327 328 329 330 331
cyclic_params(1)= NO

[ENERGY]
electrostatic   = PME                         # [CUTOFF,PME]
switchdist      = 7.5                         # switch distance
cutoffdist      = 8.0                         # cutoff distance
pairlistdist    = 9.0                         # pair-list distance
table_order     = 1                           # order of lookup table
pme_ngrid_x     = 64                          # grid size_x in [PME]
```

```
pme_ngrid_y     = 64                            # grid size_y in [PME]
pme_ngrid_z     = 64                            # grid size_z in [PME]

[DYNAMICS]
integrator      = LEAP                          # [LEAP,VVER]
nsteps          = 1500000                       # number of MD steps
timestep        = 0.002                         # time step (ps)
eneout_period   = 1000                          # energy output period
crdout_period   = 1000                          # coordinates output period
nbupdate_period = 5                             # nonbond update period
rstout_period   = 1500000                       # restart output period

[CONSTRAINTS]
rigid_bond      = YES                           # constraints all bonds
                                                # involving hydrogen

[ENSEMBLE]
ensemble        = NVT                           # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                      # thermostat
temperature     = 300.0                         # initial and target
                                                # temperature (K)

[BOUNDARY]
type            = PBC                           # [NOBC,PBC]
box_size_x      = 64.0                          # box size (x) in [PBC]
box_size_y      = 64.0                          # box size (y) in [PBC]
box_size_z      = 64.0                          # box size (z) in [PBC]
```

16.5 Analysis of REMD simulation

In this step, we examine the performance of REMD simulation by analyzing the standard output file, *logfile* (.log) and *dcdfile* (.dcd). We calculate (1) a time series of replica exchange, (2) a time series of temperature exchange of three arbitrary chosen replicas, and (3) a time series of total potential energy of three arbitrary chosen replicas.

First of all, we use *remd_convert* program which is one of the post-processing programs in the **GENESIS** package. **remd_convert** is an utility to extract various quantities from a REMD trajectory:

```
# change to the analysis directory
$ cd tutorial/alad_remd/5_analysis/
# process the REMD trajectory with remd_convert
$ remd_convert Inp_remd_conv | tee Inp_remd_conv.log
```

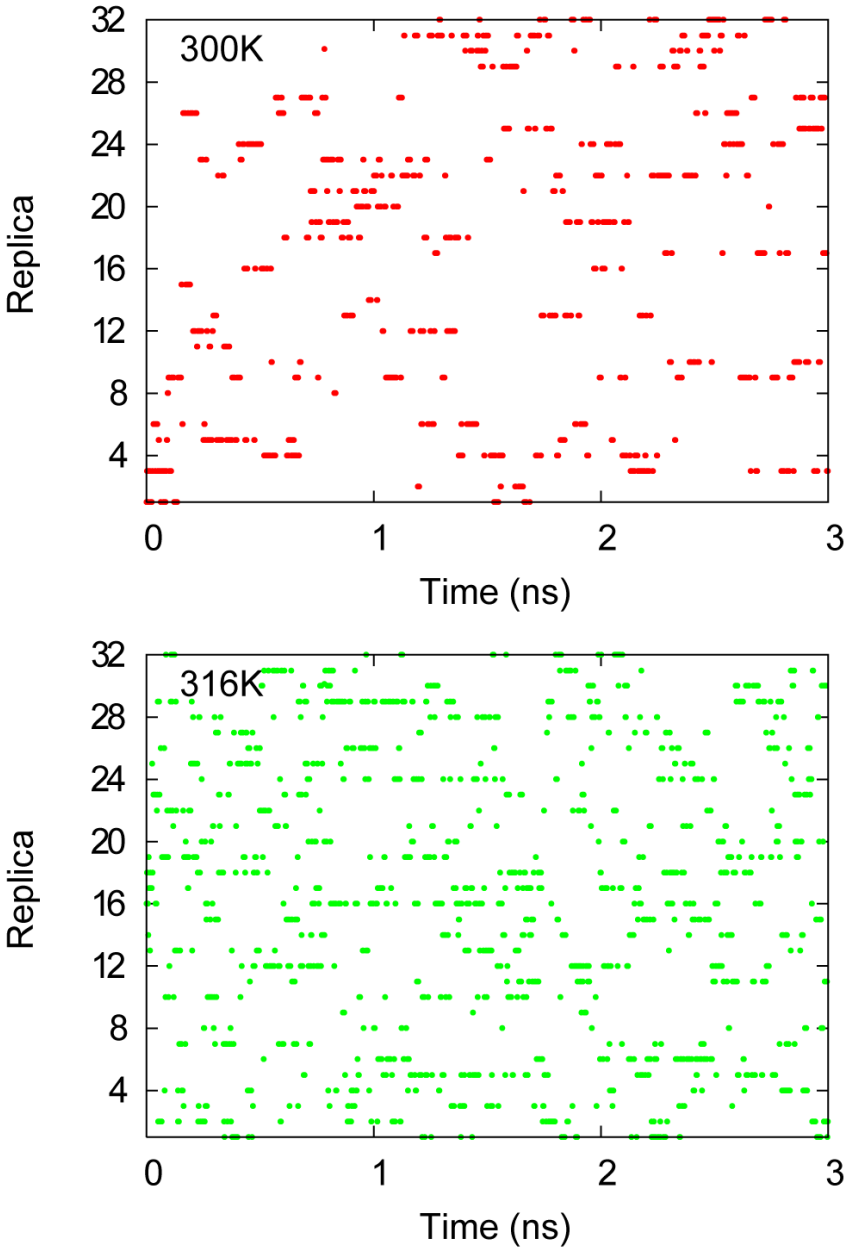
The *control file* for **remd_convert** (*Inp_remd_conv*) are shown below. In **[INPUT]** section, we specify the output files (`dcdfile` and `remfile`) from the REMD simulation. In **[SELECTION]** section, we define a group (`group1`) of the heavy atoms of the ALAD which are fitted. In the **[FITTING]** section, the fitting method is set. In this case, the translations `TR` and rotations `ROT` are allowed for the fitting.

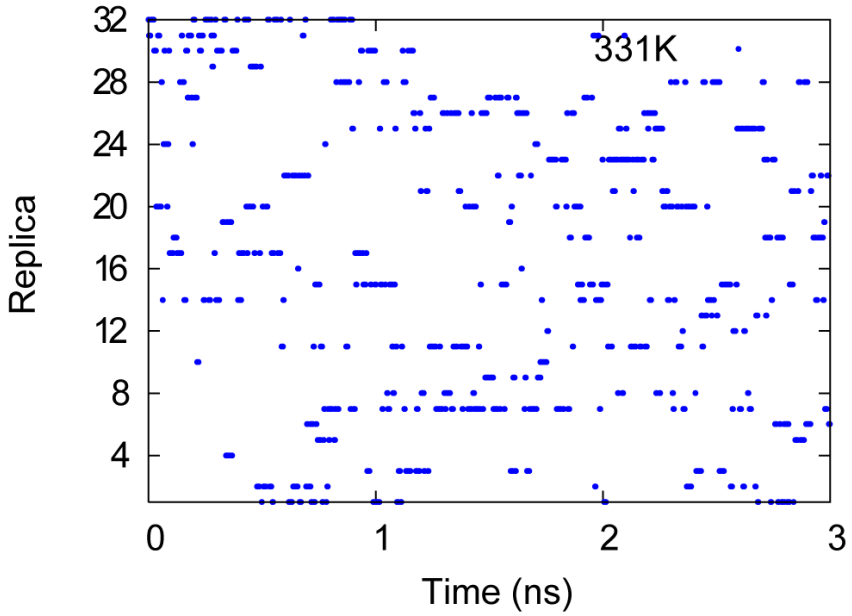
In the **[OPTION]** section, the type of conversion is set to `convert_type=PARAMETER`, and the indices of the type to be converted are set to `convert_ids=1 17 32`. Period of the `dcdfile` trajectory and the output format are set to `dcd_md_period=1000` and `trjout_format=DCD`, respectively. The selection of output atoms is set to `trjout_atom=1`, and a periodic boundary correction is set to `pbc_correct=MOLECULE`.

```
[INPUT]
psffile = ../1_setup/ala.psf                  # protein structure file
reffile = ../1_setup/ala.pdb                  # PDB file
dcdfile = ../4_production/remd_run{}.dcd
remfile = ../4_production/remd_run{}.rem

[OUTPUT]
pdbfile = ./remd_run_param.pdb                 # PDB file
trjfile = ./remd_run_param{}.trj               # trajectory file

[SELECTION]
group1          = molname:alad and heavy       # selection group 1
mole_name1      = alad PROT:1:ALAD PROT:1:ALAD

[FITTING]
fitting_method  = TR+ROT                       # method
fitting_atom    = 1                            # atom group
zrot_ngrid      = 10                           # number of z-rot grids
zrot_grid_size  = 1.0                          # z-rot grid size

[OPTION]
check_only      = NO                           # (YES/NO)
convert_type    = PARAMETER                    # (REPLICA/PARAMETER)
convert_ids     = 1 17 32
dcd_md_period   = 1000                         # input dcdfile MD period
trjout_format   = DCD                          # (PDB/DCD)
trjout_type     = COOR                         # (COOR/COOR+BOX)
trjout_atom     = 1                            # selection group output
pbc_correct     = MOLECULE                     # (NO/MOLECULE)
```

To examine (1) the time series of replica exchange, we use the standard output file of the REMD simulation to extract time series of *ReplicaID*:

```
$  grep ' ReplicaID :'  ./remd_run.out |
   awk '/  ReplicaID  :/{print $3, $19, $34}'> ./remd_run.replica
$ gnuplot
gnuplot> set xlabel "Time (ns)"
gnuplot> set ylabel "Replica"
gnuplot> plot "remd_run.replica" u ($0*0.002):1 w p pt 7 ps 0.5 lt 1
         title "300K"
gnuplot> plot "remd_run.replica" u ($0*0.002):2 w p pt 7 ps 0.5 lt 2
         title "316K"
gnuplot> plot "remd_run.replica" u ($0*0.002):3 w p pt 7 ps 0.5 lt 3
         title "331K"
```

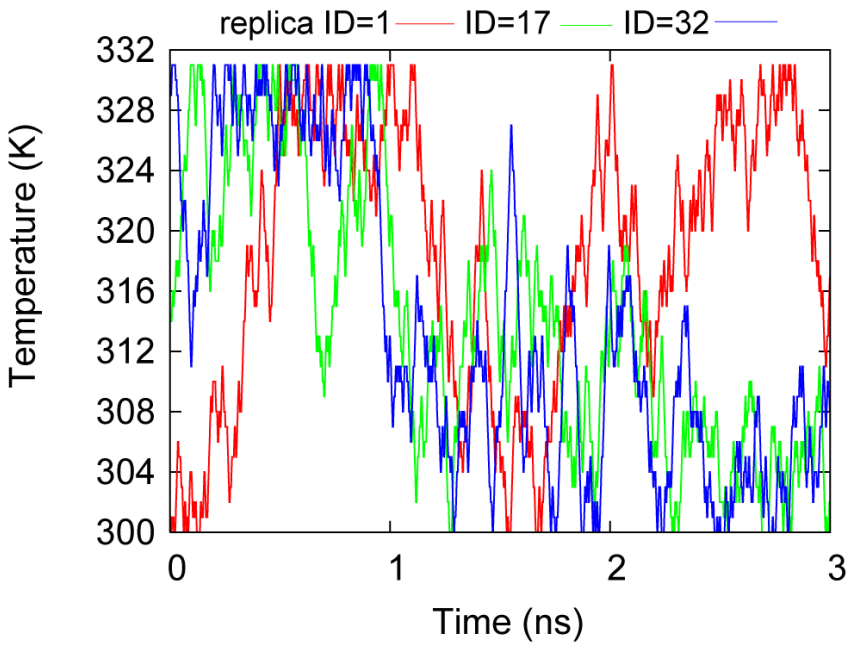

300K
316K
Replica
Time (ns)
32
28
24
20
16
12
8
4
0
1
2
3



We can observed random walks in replica space at `parameterID=1` (300 K), `17` (316 K) and `32` (331 K) colored by red, green and blue, respectively.

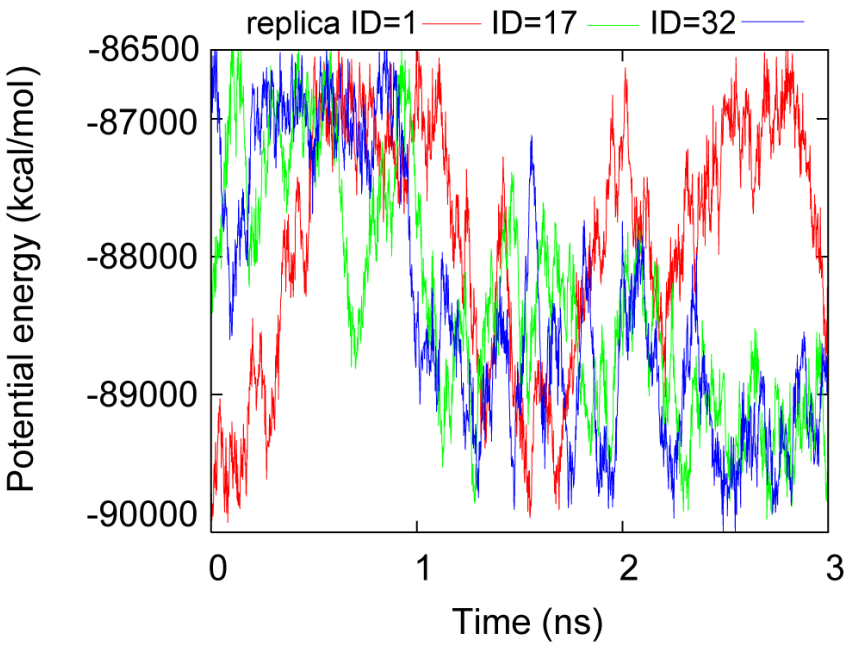
To examine (2) the time series of temperature exchange of three arbitrary chosen replicas (replica id = 1, 17 and 32), we use the standard output file of the REMD simulation to extract time series of *Parameter*:

```
$ grep ' Parameter  :' ./remd_run.param |
  awk '/  Parameter  :/{print $3, $19, $34}'> ./remd_run.param
$ gnuplot
gnuplot> set xlabel "Time (ns)"
gnuplot> set ylabel "Temperature [K]"
gnuplot> plot "remd_run.param" u ($0*0.002):1 w l lw 2 lt 1
               title "replicaID=1",\
              "remd_run.param" u ($0*0.002):2 w l lw 2 lt 2
               title "replicaID=17",\
              "remd_run.param" u ($0*0.002):3 w l lw 2 lt 3
               title "replicaID=32"
```



To examine (3) the time series of total potential energy of three arbitrary chosen replicas (replica id = 1, 17 and 32), we use the `logfile` of the REMD simulation to extract time series of *POTENTIAL_ENE*:

```
$ grep 'INFO:' remd_run01.log | tail -n +2 | awk '{print $5}'>
  ./remd_run01.ene
$ grep 'INFO:' remd_run17.log | tail -n +2 | awk '{print $5}'>
  ./remd_run17.ene
$ grep 'INFO:' remd_run32.log | tail -n +2 | awk '{print $5}'>
  ./remd_run32.ene
$ gnuplot
gnuplot> set xlabel "Time (ns)"
gnuplot> set ylabel "Potential energy (kcal/mol)"
gnuplot> plot "remd_run01.ene" u ($0*0.002):1 w l lw 1.5 lt 1
               title "replicaID=1",\
              "remd_run17.ene" u ($0*0.002):1 w l lw 1.5 lt 1
               title "replicaID=17",\
              "remd_run32.ene" u ($0*0.002):1 w l lw 1.5 lt 1
               title "replicaID=32"
```



We can observe random walks in temperature and potential energy spaces at `replicaID=1`, `17` and `32`, colored by red, green and blue, respectively. These results indicate that the REMD simulation performed satisfactorily.

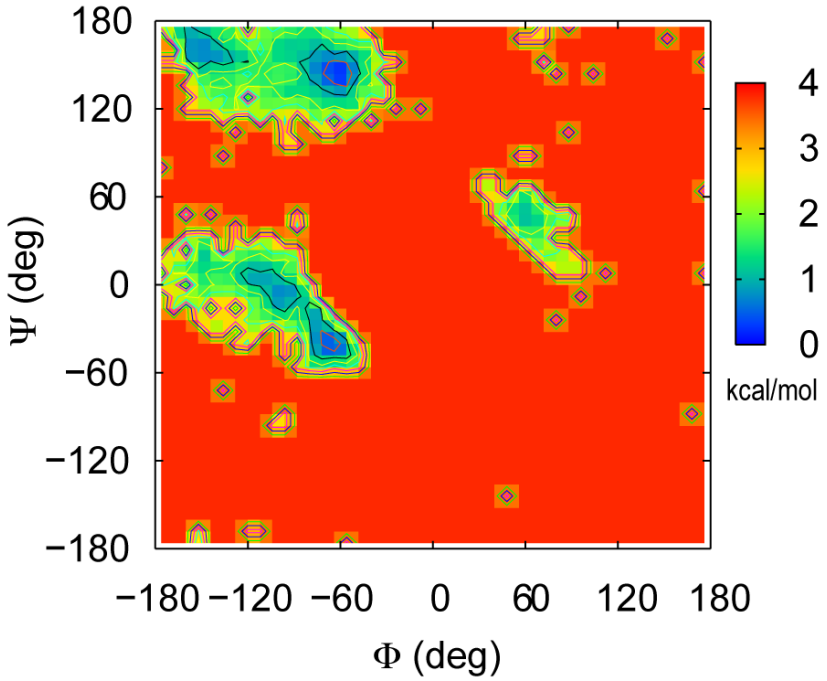
Finally we plot PMF (Potential of Mean Force) surface versus two torsion angles: CLP-NL-CA-CRP (Φ) and NL-CA-CRP-NR (Ψ).

```
# generate ctrl file for trj_analysis to calculate torsion angles
$ trj_analysis -h ctrl > Inp_remd_torsion
# calculate torsion angles with trj_analysis
$ trj_analysis Inp_remd_torsion | tee Inp_remd_torsion.log
```

The *control file* for trj_analysis (*Inp_remd_torsion*) are shown below. In **[OPTION]** section, we specify four atom names which are consisting torsion angle. Each atom name is defined using the segment name and the residue name.

```
[INPUT]
psffile = ../1_setup/alad.psf                    # protein structure file
reffile = ../1_setup/alad.pdb                    # PDB file

[OUTPUT]
torfile = ./remd_run_param01.tor                 # torsion file

[TRAJECTORY]
trjfile1          = ./remd_run_param01.trj       # trajectory file
md_step1          = 1500000                      # number of MD steps
mdout_period1     = 1000                         # MD output period
ana_period1       = 1                            # analysis period
repeat1           = 1

trj_format        = DCD                          # (PDB/DCD)
trj_type          = COOR                         # (COOR/COOR+BOX)
trj_natom         = 10                           # number of atoms
                                                 # in trajectory file


[OPTION]
```

```
check_only        = NO                              # (YES/NO)
torsion1          = PROT:1:ALAD:CLP PROT:1:ALAD:NL \
                    PROT:1:ALAD:CA  PROT:1:ALAD:CRP
torsion2          = PROT:1:ALAD:NL  PROT:1:ALAD:CA \
                    PROT:1:ALAD:CRP PROT:1:ALAD:NR
```



CHAPTER

SEVENTEEN

TUTORIAL 4: REUS SIMULATION OF ALANINE DIPEPTIDE

In this chapter, a tutorial for a replica-exchange umbrella sampling molecular dynamics (REUS) simulation with **GENESIS** is provided. The aim is to introduce a new **GENESIS** user how to perform REUS simulation of alanine dipeptide (ALAD) in water. We recommend, the new user should try *Tutorial 3: REMD Simulation of Alanine Dipeptide* before, as the procedures of performing the simulation are simplier.

The input files of this chapter are in `tutorial/alad_reus/` of the *GENESIS* package. Before you start the tutorial, please note the following things:

- Running all of the simulations requires several days.
- *VMD* [17] is required for building a simulation system and the visualization of results.
- *gnuplot* [45] is required for making plots.
- PATH environment variable includes a directory with **GENESIS** binary.

If you are a Mac OSX user, it is recommended to make an alias for *VMD*:

```
$ alias vmd="/Applications/VMD\ 1.9.app/Contents/MacOS/startup.command"
```

`tutorial/alad_reus/1_setup` directory includes *topfile* (topology file), *parfile* (parameter file), *psffile* (PSF file), *pdbfile* (initial structure) required for the REUS simulation of ALAD.

We skip *Minimization* and *Equilibration* sections required to perform before the REUS simulation, as these procedures are already described in *Tutorial 3: REMD Simulation of Alanine Dipeptide*. Please proceed to the next section after completing minimization and equilibration of your system as in *Tutorial 3: REMD Simulation of Alanine Dipeptide*.

17.1 Replica temperatures and umbrella potentials

To perform REUS simulation we need to set the type of exchanged variables. In this case we use temperature and restraint potentials of dihedral angle.

1. To choose replica temperatures, we recommend to use *Temperature generator for REMD-simulations* [50]. This web server automatically generate number of replicas and their temperatures according to provided information. In this case, we use four temperatures of 300K, 301K, 302K and 303K.

2. To choose umbrella potentials, a user needs to find appropriate reaction coordinates describing the conformational changes of the system. ALAD is a common model for backbone conformational analyses, so we use dihedral angles as the reaction coordinates.

17.2 Production simulation

Before performing REUS simulation, the system needs to be equilibrated at each temperature and each dihedral restraint. The following commands perform s 20-picosecond NVT molecular dynamics simulations.

```
# change to the production directory
$ cd tutorial/alad_reus/4_production/
# perform equilibration at each temperature and each dihedral restraint
  with ATDYN by submitting batch job script
$ qsub reus_run_eq.sh
```

The batch job script *reus_run_eq.sh* is at `tutorial/alad_reus/4_production/` for reference.

In **[OUTPUT]** section of the *control file*, *logfile* gives log file for each replica and *remfile* gives replica exchange parameter ID file. In this section "{}" returns a series of 16 output files (product of number of replicas in each dimension).

In **[REMD]** section of the *control file*, the number of dimensions is set to `dimension=2`, and the period between replica exchanges is set to `exchange_period=0` for equilibration (no need to be exchanged in this step). Type of exchanged variable in each dimension is set to `type(1)=TEMPERATURE` and `type(2)=RESTRAINT` , the number of replicas in each dimension is set to `nreplica(1)=4` and `nreplica(2)=4`. The variables in each dimension are set to `parameters(1)=300 301 302 303` and `parameters(2)= (1.0 144.0) (1.1 146.0) (1.2 148.0) (1.3 150.0)`. For `type=RESTRAINT`, force constants and reference values should be set in the parentheses. `cyclic_params(1)=NO` determines whether **ATDYN** performs replica exchanges cyclically. See *REMD* for details.

In **[DYNAMICS]** section of the *control file*, the total number of simulation steps is set to `nsteps=10000`, and the time step is set to `timestep=0.002 ps`. This results into a `nstep*timestep = 20 ps` simulation. Since the output period for coordinates is set to `crdout_period=50`, we have the total of `nsteps/crdout_period=200` snapshots in the output trajectory.

In **[SELECTION]** section of the *control file*, the groups of selected atom(s) used in **[REMD]** and **[RESTRAINTS]** sections are set. The available expressions in `group` keyword are listed in *Selection*. In this case, atom names "NL", "CA", "CRP" and "NR" of `ala.pdb` are selected for each group.

In **[RESTRAINTS]** section of the *control file*, there are keywords for external harmonic restraint functions. The restraint functions are applied to the selected atoms in the **[SELECTION]** section. In this case, a restraint potential with `constant1=1.0` and `reference1=144.0` is applied to dihedral angle NL-CA-CRP-NR of ALAD.

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ala.psf                  # protein structure file
pdbfile = ../1_setup/ala.pdb                  # PDB file
rstfile = ../3_equilibration/reus_eq2.rst     # restart file
```

```
reffile = ../1_setup/ala.pdb                 # reference for restraints

[OUTPUT]
logfile = reus_run_eq{}.log                 # log file of each replica
dcdfile = reus_run_eq{}.dcd                 # DCD trajectory file
rstfile = reus_run_eq{}.rst                 # restart file
remfile = reus_run_eq{}.rem                 # replica exchange ID file

[REMD]
dimension       = 2
exchange_period = 0

type(1)         = TEMPERATURE
nreplica(1)     = 4
parameters(1)   = 300 301 302 303
cyclic_params(1)= NO

type(2)         = RESTRAINT
nreplica(2)     = 4
parameters(2)   = (1.0 144.0) (1.1 146.0) (1.2 148.0) (1.3 150.0)
cyclic_params(2)= NO
rest_function(2)= 1

[ENERGY]
electrostatic   = PME                        # [CUTOFF,PME]
switchdist      = 7.5                        # switch distance
cutoffdist      = 8.0                        # cutoff distance
pairlistdist    = 9.0                        # pair-list cutoff distance
table_order     = 1                          # order of lookup table
pme_ngrid_x     = 64                         # grid size_x in [PME]
pme_ngrid_y     = 64                         # grid size_y in [PME]
pme_ngrid_z     = 64                         # grid size_z in [PME]

[DYNAMICS]
integrator      = LEAP                       # [LEAP,VVER]
nsteps          = 10000                      # number of MD steps
timestep        = 0.002                      # time step (ps)
eneout_period   = 50                         # energy output period
crdout_period   = 50                         # coordinates output period
nbupdate_period = 5                          # nonbond update period
rstout_period   = 10000                      # restart output period

[CONSTRAINTS]
rigid_bond      = YES                        # constraints all bonds
                                             # involving hydrogen

[ENSEMBLE]
ensemble        = NVT                        # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                   # thermostat
temperature     = 300.0                      # initial and target
                                             # temperature (K)

[BOUNDARY]
type            = PBC                        # [NOBC,PBC]
box_size_x      = 64.0                       # box size (x) in [PBC]
box_size_y      = 64.0                       # box size (y) in [PBC]
box_size_z      = 64.0                       # box size (z) in [PBC]
```

```
[SELECTION]
group1          = an:NL                         # restraint group 1
group2          = an:CA                         # restraint group 2
group3          = an:CRP                        # restraint group 3
group4          = an:NR                         # restraint group 4

[RESTRAINTS]
nfunctions      = 1                             # number of functions
function1       = DIHED                         # restraint function type
constant1       = 1.0                           # force constant
reference1      = 144.0                         # reference
select_index1   = 1 2 3 4                       # restraint groups
```

After equilibrating the system, we move on to the production simulation of REUS. The following commands perform 6-nanosecond REUS simulations in NVT ensemble:

```
# perform production simulation of REUS with ATDYN by submitting a
batch job script
$ qsub reus_run.sh
```

The batch job script *reus_run.sh* is at `tutorial/alad_reus/4_production/` for reference.

Differences from the previous equilibration step are **[INPUT]**, **[REUS]** and **[DYNAMICS]** sections.

In **[INPUT]** section of the *control file* (`reus_run.inp`), *rstfile* is set to the restart file from the equilibration. In this section "{16}" also returns a series of 16 input files.

In **[REUS]** section of the *control file*, the period between replica exchanges is set to `exchange_period=1000`.

In **[DYNAMICS]** section of the *control file*, the total number of simulation step is set to `nsteps=3000000`, and the time step is set to `timestep=0.002 ps`. This results into a `nstep*timestep = 6 ns` simulation. Since the output period for coordinates is set to `crdout_period=1000`, we have the total of `nsteps/crdout_period=3000` snapshots in the output trajectory.

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/alad.psf                 # protein structure file
rstfile = ./reus_run_eq{}.rst                 # restart file
pdbfile = ../1_setup/alad.pdb                 # PDB file

[OUTPUT]
logfile = reus_run{}.log                    # log file of each replica
dcdfile = reus_run{}.dcd                    # DCD trajectory file
rstfile = reus_run{}.rst                    # restart file
remfile = reus_run{}.rem                    # replica exchange ID file

[REMD]
dimension       = 2
exchange_period = 1000
type(1)         = TEMPERATURE
nreplica(1)     = 4
parameters(1)   = 300 301 302 303
```

```
cyclic_params(1)= NO
type(2)         = RESTRAINT
nreplica(2)     = 4
parameters(2)   = (1.0 144.0) (1.1 146.0) (1.2 148.0) (1.3 150.0)
cyclic_params(2)= NO
rest_function(2)= 1

[ENERGY]
electrostatic   = PME                          # [CUTOFF,PME]
switchdist      = 7.5                          # switch distance
cutoffdist      = 8.0                          # cutoff distance
pairlistdist    = 9.0                          # pair-list distance
table_order     = 1                            # order of lookup table
pme_ngrid_x     = 64                           # grid size_x in [PME]
pme_ngrid_y     = 64                           # grid size_y in [PME]
pme_ngrid_z     = 64                           # grid size_z in [PME]

[DYNAMICS]
integrator      = LEAP                         # [LEAP,VVER]
nsteps          = 3000000                      # number of MD steps
timestep        = 0.002                        # time step (ps)
eneout_period   = 1000                         # energy output period
crdout_period   = 1000                         # coordinates output period
rstout_period   = 3000000                      # restart output period
nbupdate_period = 5                            # nonbond update period

[CONSTRAINTS]
rigid_bond      = YES                          # constraints all bonds
                                               # involving hydrogen

[ENSEMBLE]
ensemble        = NVT                          # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                     # thermostat
temperature     = 300.0                        # initial and target
                                               # temperature (K)

[BOUNDARY]
type            = PBC                          # [NOBC,PBC]
box_size_x      = 64.0                         # box size (x) in [PBC]
box_size_y      = 64.0                         # box size (y) in [PBC]
box_size_z      = 64.0                         # box size (z) in [PBC]

[SELECTION]
group1          = an:NL                        # restraint group 1
group2          = an:CA                        # restraint group 2
group3          = an:CRP                       # restraint group 3
group4          = an:NR                        # restraint group 4

[RESTRAINTS]
nfunctions      = 1                            # number of functions
function1       = DIHED                        # restraint function type
constant1       = 1.0                          # force constant
reference1      = 144.0                        # reference
select_index1   = 1 2 3 4                      # restraint groups
```

17.3 Analysis of REUS simulation

In this step, we examine the performance of REUS simulation by analyzing the standard output file, *logfile* (.log) and *dcdfile* (.dcd). We calculate (1) a time series of replica exchange, (2) a time series of parameter exchange of three arbitrary chosen replicas, and (3) a time series of total potential energy of three arbitrary chosen replicas.

First, we use *remd_convert* program which is one of the post-processing programs in the **GENESIS** package. **remd_convert** is an utility to extract various quantities from a REUS trajectory:

```
# change to the analysis directory
$ cd tutorial/alad_reus/5_analysis/
# process of the REUS trajectory with remd_convert
$ remd_convert Inp_reus_conv | tee Inp_reus_conv.log
```

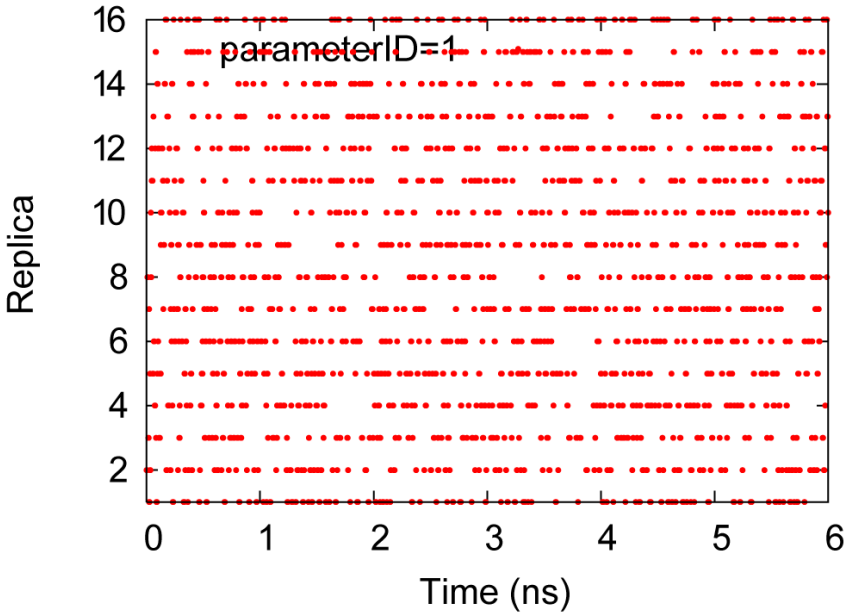
The *control file* for **remd_convert** (*Inp_reus_conv*) is shown below. In **[INPUT]** section, we specify the output files (`dcdfile` and `remfile`) from the REUS simulation. In **[SELECTION]** section, we define a group (`group1`) of the heavy atoms of ALAD which are fitted. In the **[FITTING]** section, the fitting method is set. In this case, the translations `TR` and rotations `ROT` are allowed for the fitting.

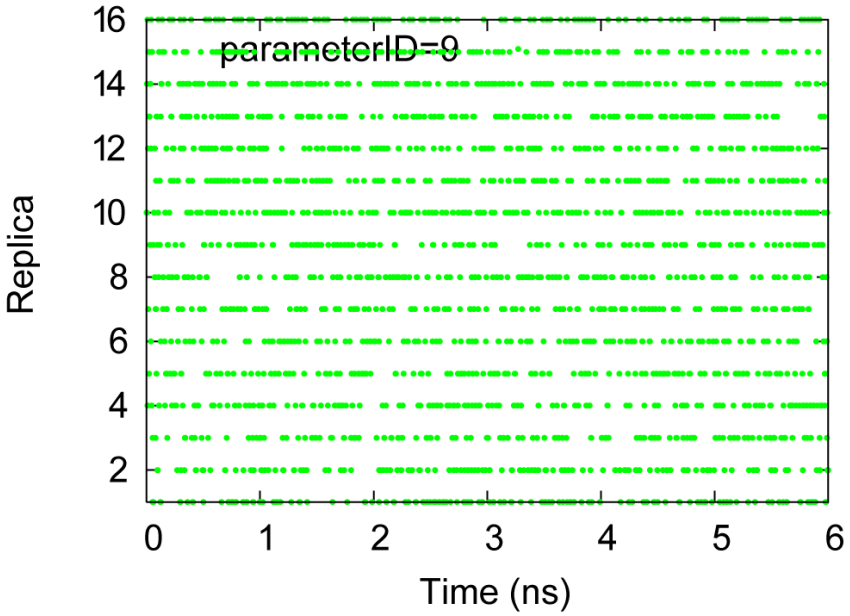
In the **[OPTION]** section, the type of conversion is set to `convert_type=PARAMETER`, and the indices of the type to be converted are set to `convert_ids=1 9 16`. Period of the `dcdfile` trajectory and the output format are set to `dcd_md_period=1000` and `trjout_format=DCD`, respectively. The selection of output atoms is set to `trjout_atom=1`, and a periodic boundary correction is set to `pbc_correct=MOLECULE`.

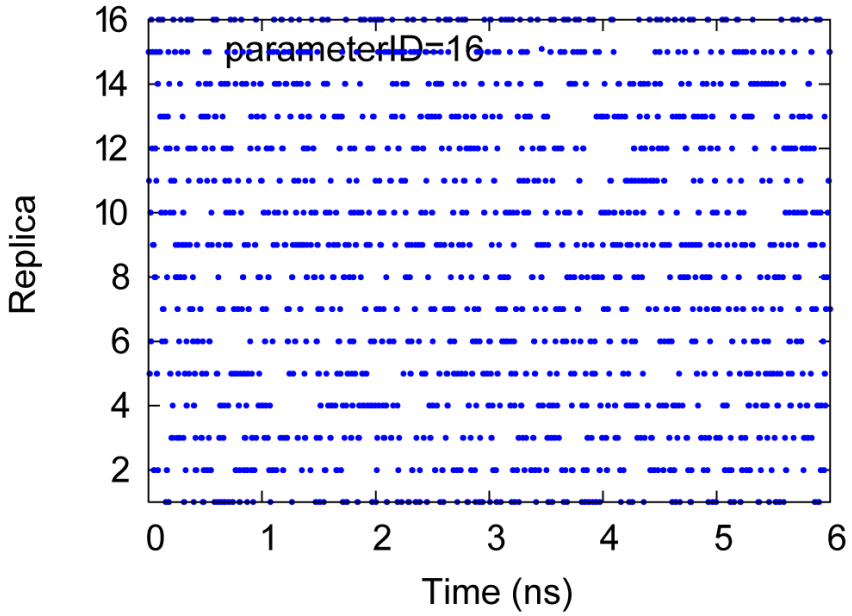
```
[INPUT]
psffile = ../1_setup/ala.psf                # protein structure file
reffile = ../1_setup/ala.pdb                # PDB file
dcdfile = ../4_production/reus_run{}.dcd
remfile = ../4_production/reus_run{}.rem

[OUTPUT]
pdbfile = ./reus_run_param.pdb              # PDB file
trjfile = ./reus_run_param{}.trj            # trajectory file

[SELECTION]
group1          = molname:alad and heavy     # selection group 1
mole_name1      = alad PROT:1:ALAD PROT:1:ALAD

[FITTING]
fitting_method  = TR+ROT                     # method
fitting_atom    = 1                          # atom group
zrot_ngrid      = 10                         # number of z-rot grids
zrot_grid_size  = 1.0                        # z-rot grid size

[OPTION]
check_only      = NO                         # (YES/NO)
convert_type    = PARAMETER                  # (REPLICA/PARAMETER)
convert_ids     = 1 9 16
dcd_md_period   = 1000                       # input dcdfile MD period
trjout_format   = DCD                        # (PDB/DCD)
trjout_type     = COOR                       # (COOR/COOR+BOX)
trjout_atom     = 1                          # selection group output
pbc_correct     = MOLECULE                   # (NO/MOLECULE)
```

To examine (1) the time series of replica exchange, we use the standard output file of the REUS simulation to extract time series of *ReplicaID*:

```
$  grep ' ReplicaID : ' ./reus_run.out |
   awk '/  ReplicaID  :/{print $3, $11, $18}'> ./reus_run.replica
$ gnuplot
gnuplot> set xlabel "Time (ns)"
gnuplot> set ylabel "Replica"
gnuplot> plot "reus_run.replica" u ($0*0.002):1 w p pt 7 ps 0.5 lt 1
               title "ParameterID=1"
gnuplot> plot "reus_run.replica" u ($0*0.002):2 w p pt 7 ps 0.5 lt 2
               title "ParameterID=9"
gnuplot> plot "reus_run.replica" u ($0*0.002):3 w p pt 7 ps 0.5 lt 3
               title "ParameterID=16"
```



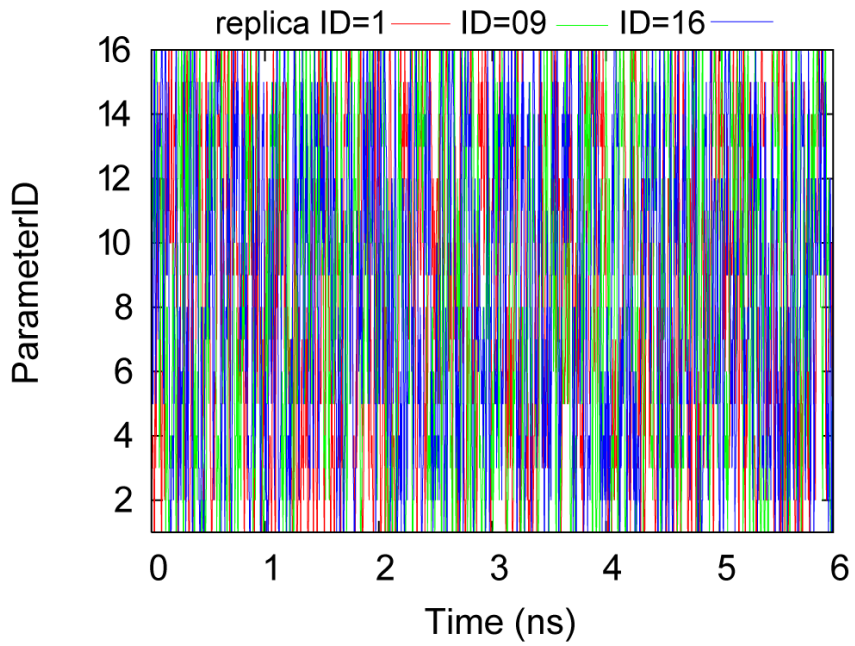




We can observed random walks in replica space at `parameterID=1` (300 K, (1.0 144.0)), `parameterID=9` (302 K, (1.0 144.0)) and `parameterID=16` (303 K, (1.3 150.0)) colored by red, green and blue, respectively.

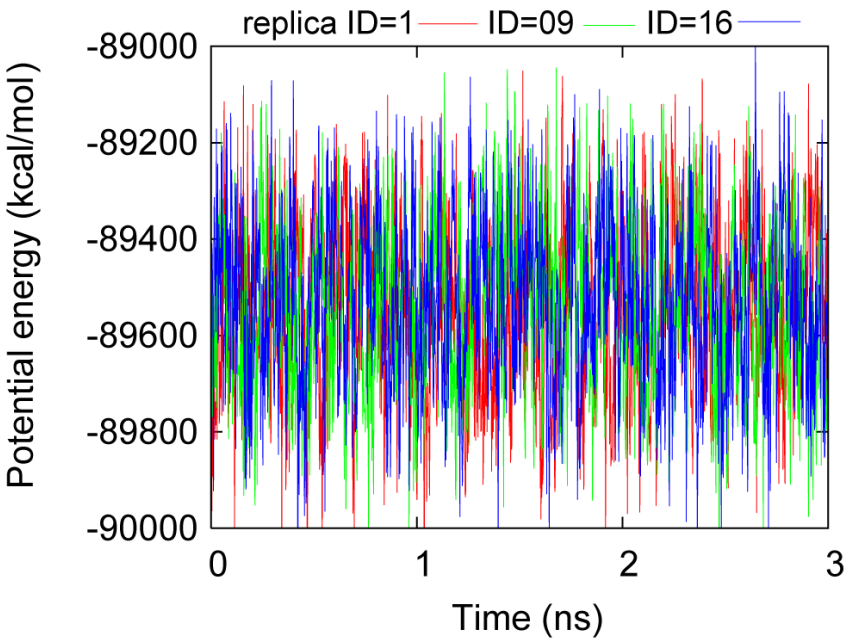
To examine (2) the time series of parameter exchange of three arbitrary chosen replicas (replica id = 1, 9 and 16), we use the standard output file of the REUS simulation to extract time series of *ParameterID*:

```
$ grep ' ParameterID  :' ./remd_run.param |
  awk '/  ParameterID :/{print $2, $10, $17}'> ./remd_run.param
$ gnuplot
gnuplot> set xlabel "Time (ns)"
gnuplot> set ylabel "Temperature [K]"
gnuplot> plot "reus_run.param" u ($0*0.002):1 w l lw 2 lt 1
               title "replicaID=1",\
              "reus_run.param" u ($0*0.002):2 w l lw 2 lt 2
               title "replicaID=9",\
              "reus_run.param" u ($0*0.002):3 w l lw 2 lt 3
               title "replicaID=16"
```



To examine (3) the time series of total potential energy of three arbitrary chosen replicas (replica id = 1, 9 and 16), we use the `enefile` of the REUS simulation to extract time series of *POTENTIAL_ENE*:

```
$ grep 'INFO:' reus_run01.log | tail -n +2 | awk '{print $5}'>
  ./reus_run01.ene
$ grep 'INFO:' reus_run09.log | tail -n +2 | awk '{print $5}'>
  ./reus_run09.ene
$ grep 'INFO:' reus_run16.log | tail -n +2 | awk '{print $5}'>
  ./reus_run16.ene
$ gnuplot
gnuplot> set xlabel "Time (ns)"
gnuplot> set ylabel "Potential energy (kcal/mol)"
gnuplot> plot "reus_run01.ene" u (\$0*0.002):1 w l lw 1.5 lt 1
               title "replicaID=1",\
              "reus_run09.ene" u (\$0*0.002):1 w l lw 1.5 lt 1
               title "replicaID=9",\
              "reus_run16.ene" u (\$0*0.002):1 w l lw 1.5 lt 1
               title "replicaID=16"
```



We can observe random walks in temperature and potential energy spaces at `replicaID=1`, `9` and `16`, colored by red, green and blue, respectively. These results indicate that the REUS simulation performed satisfactorily.

To plot PMF (Potential of Mean Force) surface versus two torsion angles: CLP-NL-CA-CRP (Φ) and NL-CA-CRP-NR (Ψ) at 300K with restraint bias removed, WHAM (Weighted Histogram Analysis Method) could be applied to the REUS trajectory. However, WHAM is beyond the scope of this tutorial.

CHAPTER

EIGHTEEN

TUTORIAL 5: PARALLEL INPUT/OUTPUT SCHEME

If a simulation system is huge, and a larger number of processors is availalbe, it takes a significant amount of time to gather data for restart and trajectory file updates. So, it is preferred, each process writes local information separately, rather than a single does for all. In **GENESIS**, the *parallel input/output (I/O)* scheme is available to deal effectively with larger restart or trajectory files.

Note: The *parallel I/O* scheme are available in **SPDYN** only.

In this tutorial, we explain an example of the *parallel I/O* with BPTI protein used in *Tutorial 1: Building and Simulating BPTI in Water*. The procedures of the simulation are identical, except that we are dealing with the *parallel I/O* using the same number of input files and the number of MPI processes.

The input files of this tutorial are at `tutorial/bpti_parallel_io/` of the *GENESIS* package. In all cases, we use 8 MPI processes.

18.1 Minimization with restraints on the protein

Minimization is done to remove unphysical steric clashes or inappropriate geometries. Thus, it is recommended to do a minimiation before doing a molecular dynamics simulation. To make use of the parallel i/o, the first step to do before minimization is to make the parallel restart files, each of which will be assigned to each MPI processor for minimization run.

18.1.1 Making restart for minimization

prst_setup is used to make parallel restart files. First, let's check the *control file* (`run.inp`) in the BPTI tutorial.

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ionize.psf               # protein structure file
pdbfile = ../1_setup/ionize.pdb               # PDB file
reffile = ../1_setup/ionize.pdb               # reference for restraints

[OUTPUT]
dcdfile = run.dcd                             # DCD trajectory file
```

```
rstfile = run.rst                                # restart file

[ENERGY]
electrostatic   = PME                            # [CUTOFF,PME]
switchdist      = 10.0                           # switch distance
cutoffdist      = 12.0                           # cutoff distance
pairlistdist    = 13.5                           # pair-list cutoff distance
table           = YES                            # usage of lookup table
table_order     = 0                              # order of lookup table
pme_ngrid_x     = 72                             # grid size_x in [PME]
pme_ngrid_y     = 80                             # grid size_y in [PME]
pme_ngrid_z     = 72                             # grid size_z in [PME]

[MINIMIZE]
nsteps          = 1000                           # number of steps
eneout_period   = 100                            # energy output period
crdout_period   = 100                            # coordinates output period
rstout_period   = 1000                           # restart output period

[BOUNDARY]
type            = PBC                            # [PBC,NOBC]
box_size_x      = 70.8250                        # box size (x) in [PBC]
box_size_y      = 83.2579                        # box size (y) in [PBC]
box_size_z      = 69.0930                        # box size (z) in [PBC]

[SELECTION]
group1          = backbone                       # index of restraint group 1

[RESTRAINTS]
nfunctions      = 1                              # number of functions
function1       = POSI                           # restraint function type
constant1       = 10.0                           # force constant
select_index1   = 1                              # restraint group
```

The *control file* is very similar to this except a few changes. First, in **[OUTPUT]** section, instead of output restart file, parallel restart file for minimization should be specified. Second, you do not need **[MINIMIZE]** section (you can just leave the same information written in the control file of the minimization). Third, you should specify the number of domains in each direction or the total number of domains (which is identical to the number of MPI processes).

Below we write the input (`setup.inp`) to generate multiple restart files for minimization. It is difficult to distinguish the necessary keywords from unnecessary ones. So, we wrote all the keywords identically to *control file* for minimization. It is necessary to change in **[OUTPUT]** and `[BOUNDARY]` sections only. In **[BOUNDARY]** section, you need to set the number of sub-domains in each direction using `domain_x`, `domain_y`, and `domain_z`, or total number of sub-domains using `domain_xyz`. In this example, we set the number of sub-domains in each domain:

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ionize.psf               # protein structure file
pdbfile = ../1_setup/ionize.pdb               # PDB file
reffile = ../1_setup/ionize.pdb               # reference for restraints

[OUTPUT]
rstfile = setup().rst                         # parallel restart file
```

```
cachepath = ./cache

[ENERGY]
electrostatic   = PME                           # [CUTOFF,PME]
switchdist      = 10.0                          # switch distance
cutoffdist      = 12.0                          # cutoff distance
pairlistdist    = 13.5                          # pair-list cutoff distance
table_order     = 0                             # order of lookup table
pme_ngrid_x     = 72                            # grid size_x in [PME]
pme_ngrid_y     = 80                            # grid size_y in [PME]
pme_ngrid_z     = 72                            # grid size_z in [PME]

[MINIMIZE]
nsteps          = 1000                          # number of steps
eneout_period   = 100                           # energy output period
crdout_period   = 100                           # coordinates output period
rstout_period   = 1000                          # restart output period

[BOUNDARY]
type            = PBC                           # [PBC,NOBC]
box_size_x      = 70.8250                       # box size (x) in [PBC]
box_size_y      = 83.2579                       # box size (y) in [PBC]
box_size_z      = 69.0930                       # box size (z) in [PBC]
domain_x        = 2
domain_y        = 2
domain_z        = 2

[SELECTION]
group1          = backbone                      # index of restraint group 1

[RESTRAINTS]
nfunctions      = 1                             # number of functions
function1       = POSI                          # restraint function type
constant1       = 10.0                          # force constant
select_index1   = 1                             # restraint group
```

Another noticeable change is `cachepath` in **[OUTPUT]** section. It is a path to a directory where intermediate files are written before generating final restart files. If you do not specify `cachepath`, all the intermediate files will be saved in memory only, and it takes more time.

Note: **rst_setup** is OpenMP parallized only (no MPI parallization).

The following commands makes multiply restart files for minimization, using 2 OpenMP threads:

```
# change to the minimization directory
$ cd tutorial/bpti_parallel_io/2_minimization/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=2
# make multiple restart files
$ rst_setup setup.inp | tee setup.out
```

After executing these commands, you generate 8 restart files named `setup0.rst` ~ `setup7.rst`. These files will be used instead of PDB and PSF files for minimization.

18.1.2 Minimization with restraint

After making restart files for minimization, you can minimize with 8 MPI processes, as the number of restart files is 8. Each MPI process read one corresponding restart file (i.e., MPI rank 0 reads `setup0.rst`).

The control input of minimization with the *parallel I/O* is very similar to the one without it:

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
rstfile = setup().rst

[OUTPUT]
dcdfile = run().dcd                           # DCD trajectory file
rstfile = run().rst                           # restart file

[ENERGY]
electrostatic   = PME                         # [CUTOFF,PME]
switchdist      = 10.0                        # switch distance
cutoffdist      = 12.0                        # cutoff distance
pairlistdist    = 13.5                        # pair-list cutoff distance
table_order     = 0                           # order of lookup table
pme_ngrid_x     = 72                          # grid size_x in [PME]
pme_ngrid_y     = 80                          # grid size_y in [PME]
pme_ngrid_z     = 72                          # grid size_z in [PME]

[MINIMIZE]
nsteps          = 1000                        # number of steps
eneout_period   = 100                         # energy output period
crdout_period   = 100                         # coordinates output period
rstout_period   = 1000                        # restart output period

[BOUNDARY]
type            = PBC                         # [PBC,NOBC]
box_size_x      = 70.8250                     # box size (x) in [PBC]
box_size_y      = 83.2579                     # box size (y) in [PBC]
box_size_z      = 69.0930                     # box size (z) in [PBC]
domain_x        = 2
domain_y        = 2
domain_z        = 2

[SELECTION]
group1          = backbone                    # index of restraint group 1

[RESTRAINTS]
nfunctions      = 1                           # number of functions
function1       = POSI                        # restraint function type
constant1       = 10.0                        # force constant
select_index1   = 1                           # restraint group
```

The main difference between this and control input of minimization without the *parallel I/O* is at the keywords of input and output files. If there are multiple files corresponding to each MPI process, `()` should be added as above.

You can perform minimization with the following command:

```
$ mpirun -np 8 spdyn run.inp | tee run.out
```

The outputs of the *parallel I/O* is identical to the ones without it.

18.2 Heat-up with restraints on the protein

The purpose of this procedure is explained in *Tutorial 1: Building and Simulating BPTI in Water*. Here, we just explain how to use the *parallel I/O*.

For the task, first you need to change to the heat-up directory:

```
# change to the heat-up directory
$ cd tutorial/bpti_parallel_io/3_heating/
```

18.2.1 Restart files for heat-up simulation

As mentioned in the previous section, we may think that same control input by adding `()` is enough to set up the *parallel I/O* simulation.

The control input file, corresponding to the one in *Tutorial 1: Building and Simulating BPTI in Water*, can be written as follow by just by replacing input and output file names:

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
rstfile = ../2_minimization/run().rst         # restart file

[OUTPUT]
dcdfile = run().dcd                           # DCD trajectory file
rstfile = run().rst                           # restart file

[ENERGY]
electrostatic   = PME                         # [CUTOFF,PME]
switchdist      = 10.0                        # switch distance
cutoffdist      = 12.0                        # cutoff distance
pairlistdist    = 13.5                        # pair-list cutoff distance
table_order     = 1                           # order of lookup table
pme_ngrid_x     = 72                          # grid size_x in [PME]
pme_ngrid_y     = 80                          # grid size_y in [PME]
pme_ngrid_z     = 72                          # grid size_z in [PME]

[DYNAMICS]
integrator      = LEAP                        # [LEAP,VVER]
nsteps          = 5000                        # number of MD steps
timestep        = 0.002                       # time step (ps)
eneout_period   = 50                          # energy output period
crdout_period   = 50                          # coordinates output period
rstout_period   = 5000                        # restart output period
annealing       = YES                         # simulated annealing
anneal_period   = 50                          # annealing period
dtemperature    = 3                           # temperature change at
                                              # annealing (K)
```

```
[CONSTRAINTS]
rigid_bond      = YES                           # constraints of all bonds

[ENSEMBLE]
ensemble        = NVT                           # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                      # thermostat
temperature     = 0.1                           # initial temperature (K)

[BOUNDARY]
type            = PBC                           # [PBC,NOBC]
domain_x        = 2
domain_y        = 2
domain_z        = 2

[SELECTION]
group1          = backbone                      # index of restraint group 1

[RESTRAINTS]
nfunctions      = 1                             # number of functions
function1       = POSI                          # restraint function type
constant1       = 10.0                          # force constant
select_index1   = 1                             # restraint group
```

However, if you use this control input, you get the following error message:

```
Pio_Check_Compatible> ERROR : [Constraint]rigid-bond: must be NO
Pio_Check_Compatible> ERROR : [Ensemble]ensemble: must be NVE
Pio_Check_Compatible> WARNING:[Boundary]box size: t=0 : 70.12 82.56 68.39
Pio_Check_Compatible> ERROR  rank_no =     0
```

This is due to **[CONSTRAINTS]** section, which does not exist in the previous control input file. Unlike the restart files without the *parallel I/O*, the potential function information is included in the restart files (that's why PSF files is not necessary for the *parallel I/O*). So any change of potential functions requires a consistent restart files to be regenerated.

The procedure of regenerating restart files is very similar to the case of minimization. The control input to generate restart files is as follow:

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
psffile = ../1_setup/ionize.psf               # protein structure file
pdbfile = ../1_setup/ionize.pdb               # PDB file
reffile = ../1_setup/ionize.pdb               # reference for restraints
rstfile = ../2_minimization/run().rst         # restart file

[OUTPUT]
rstfile = setup().rst                         # parallel restart file
cachepath = ./cache

[OUTPUT]
dcdfile = run().dcd                           # DCD trajectory file
rstfile = run().rst                           # restart file

[ENERGY]
```

```
electrostatic   = PME                           # [CUTOFF,PME]
switchdist      = 10.0                          # switch distance
cutoffdist      = 12.0                          # cutoff distance
pairlistdist    = 13.5                          # pair-list cutoff distance
table_order     = 1                             # order of lookup table
pme_ngrid_x     = 72                            # grid size_x in [PME]
pme_ngrid_y     = 80                            # grid size_y in [PME]
pme_ngrid_z     = 72                            # grid size_z in [PME]

[DYNAMICS]

[CONSTRAINTS]
rigid_bond      = YES                           # constraints of all bonds

[ENSEMBLE]
ensemble        = NVT                           # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                      # thermostat

[BOUNDARY]
type            = PBC                           # [PBC,NOBC]
box_size_x      = 70.8250                       # box size (x) in [PBC]
box_size_y      = 83.2579                       # box size (y) in [PBC]
box_size_z      = 69.0930                       # box size (z) in [PBC]
domain_x        = 2
domain_y        = 2
domain_z        = 2

[SELECTION]
group1          = backbone                      # index of restraint group 1

[RESTRAINTS]
nfunctions      = 1                             # number of functions
function1       = POSI                          # restraint function type
constant1       = 10.0                          # force constant
select_index1   = 1                             # restraint group
```

In this case, the restart files generated after minimization are in **[INPUT]** section. Here, we generated restart files without setting the number of sub-domains in each direction, but setting the total number of processes only with the keyword `domain_xyz`.

After running `prst_setup setup.inp | tee setup.out`, the restart files for the heat-up simulation are regenerated. Except **[INPUT]**, **[OUTPUT]**, and **[BOUNDARY]** sections.

18.2.2 Making restart files for heat-up simulation

Similarly to minimization, `setup0.rst` ~ `setup7.rst` files are used as restart files for the simulation. The control file for the simulations is:

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
rstfile = setup().rst                         # restart file

[OUTPUT]
dcdfile = run().dcd                           # DCD trajectory file
```

```
rstfile = run().rst                          # restart file

[ENERGY]
electrostatic   = PME                        # [CUTOFF,PME]
switchdist      = 10.0                       # switch distance
cutoffdist      = 12.0                       # cutoff distance
pairlistdist    = 13.5                       # pair-list cutoff distance
table_order     = 1                          # order of lookup table
pme_ngrid_x     = 72                         # grid size_x in [PME]
pme_ngrid_y     = 80                         # grid size_y in [PME]
pme_ngrid_z     = 72                         # grid size_z in [PME]

[DYNAMICS]
integrator      = LEAP                       # [LEAP,VVER]
nsteps          = 5000                       # number of MD steps
timestep        = 0.002                      # time step (ps)
eneout_period   = 50                         # energy output period
crdout_period   = 50                         # coordinates output period
rstout_period   = 5000                       # restart output period
annealing       = YES                        # simulated annealing
anneal_period   = 50                         # annealing period
dtemperature    = 3                          # temperature change at

[CONSTRAINTS]
rigid_bond      = YES                        # constraints of all bonds

[ENSEMBLE]
ensemble        = NVT                        # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                   # thermostat
temperature     = 0.1                        # initial temperature (K)

[BOUNDARY]
type            = PBC                        # [PBC,NOBC]
domain_x        = 2
domain_y        = 2
domain_z        = 2

[SELECTION]
group1          = backbone                   # index of restraint group 1

[RESTRAINTS]
nfunctions      = 1                          # number of functions
function1       = POSI                       # restraint function type
constant1       = 10.0                       # force constant
select_index1   = 1                          # restraint group
```

By running `mpirun -np 8 spdyn run.inp | tee run.out`, the procedure of *Heat-up with restraints on the protein* could be performed. Unlike the control in the previous section of minimization, some informations in the **[BOUNDARY]** section are missing. This is due to the fact the information of the **[BOUNDARY]** section is already saved in the parallel restart files, setup0.rst~setup7.rst.

18.3 Equilibration

To do equilibration, the following command is necessary:

```
# change to the equilibration directory
$ cd tutorial/bpti_parallel_io/4_equilibration/
```

In this step, there is a change in **[ENSEMBE]** section (from NVT to NPT). Thus, we need to regenerate RST files accordingly to the change:

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
rstfile = ../3_heating/run().rst              # restart file

[OUTPUT]
rstfile = setup().rst                         # restart file

[ENERGY]
electrostatic   = PME                         # [CUTOFF,PME]
switchdist      = 10.0                        # switch distance
cutoffdist      = 12.0                        # cutoff distance
pairlistdist    = 13.5                        # pair-list cutoff distance
pme_ngrid_x     = 72                          # grid size_x in [PME]
pme_ngrid_y     = 80                          # grid size_y in [PME]
pme_ngrid_z     = 72                          # grid size_z in [PME]

[DYNAMICS]
integrator      = LEAP                        # [LEAP,VVER]
nsteps          = 5000                        # number of MD steps
timestep        = 0.002                       # time step (ps)
eneout_period   = 50                          # energy output period
crdout_period   = 50                          # coordinates output period
rstout_period   = 5000                        # restart output period

[CONSTRAINTS]
rigid_bond      = YES                         # constraints of all bonds

[ENSEMBLE]
ensemble        = NPT                         # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                    # thermostat and barostat

[BOUNDARY]
type            = PBC                         # [PBC,NOBC]
domain_x        = 2
domain_y        = 2
domain_z        = 2
```

From this input with **prst_setup**, multiple restart files are generated. After running `prst_setup setup.inp | tee setup.out`, the *parallel I/O* files are regenerated and these are used for the equilibration. In this case, we set the number of sub-domains to 2 in each dimension.

The control of the simulation run is as follow:

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
rstfile = ./setup().rst                       # restart file

[OUTPUT]
```

```
dcdfile = run().dcd                          # DCD trajectory file
rstfile = run().rst                          # restart file

[ENERGY]
electrostatic   = PME                        # [CUTOFF,PME]
switchdist      = 10.0                       # switch distance
cutoffdist      = 12.0                       # cutoff distance
pairlistdist    = 13.5                       # pair-list cutoff distance
pme_ngrid_x     = 72                         # grid size_x in [PME]
pme_ngrid_y     = 80                         # grid size_y in [PME]
pme_ngrid_z     = 72                         # grid size_z in [PME]

[DYNAMICS]
integrator      = LEAP                       # [LEAP,VVER]
nsteps          = 5000                       # number of MD steps
timestep        = 0.002                      # time step (ps)
eneout_period   = 50                         # energy output period
crdout_period   = 50                         # coordinates output period
rstout_period   = 5000                       # restart output period

[CONSTRAINTS]
rigid_bond      = YES                        # constraints all bonds
                                             # involving hydrogen

[ENSEMBLE]
ensemble        = NPT                        # [NVE,NVT,NPT]
tpcontrol       = LANGEVIN                   # barostat and thermostat
temperature     = 300.0                      # initial and target
                                             # temperature (K)
pressure        = 1.0                        # target pressure (atm)

[BOUNDARY]
type            = PBC                        # [PBC,NOBC]
```

As in the previous heat-up simulation, we skipped the simulation box size information in **[BOUNDARY]** section, because the informations is already in the restart files.

18.4 Production simulation

The control input of a production simulation is identical to that of equilibration, except the number of simulation steps and output periods in **[DYNAMICS]** section. In this case, we don't have to regenerate restart files with **prst_setup**.

The control input is almost identical to the case without the *parallel I/O*, except **[INPUT]** and **[OUTPUT]** sections. As it is already mentioned in the previous sections, `()` should be appended to file names for the *parallel I/O*. Here, control input of `[INPUT]` and `[OUTPUT]` sections only is given:

```
[INPUT]
topfile = ../1_setup/top_all27_prot_lipid.rtf # topology file
parfile = ../1_setup/par_all27_prot_lipid.prm # parameter file
rstfile = ../4_equilibration/run().rst        # restart file

[OUTPUT]
dcdfile = run().dcd                          # DCD trajectory file
```

```
rstfile = run().rst                                # restart file
```

The commands for the production simulation are:

```
# change to the production directory
$ cd tutorial/bpti_paralell_io/5_production/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=1
# perform production with SPDYN by using 8 MPI processes
$ mpirun -np 8 spdyn run.inp | tee run.out
```

18.5 Analysis: RMSD calculation

This is a procedure to compute RMSD with respect to the crystal structure to explore stability.

For analysis with the *parallel I/O* scheme we use **pcrd_conver** in stead of **crd_conver**. The command to calculate RMSDs from the production trajectory is:

```
# change to the analysis directory
$ cd tutorial/bpti_parallel_io/6_analysis/
# set the number of OpenMP threads
$ export OMP_NUM_THREADS=1
# perform analysis with pcrd_convert
$ pcrd_convert run.inp | tee run.out
```

As in the case of production simulation, the *control file* of the *parallel I/O* is almost identical to the one without it, except **[TRAJECTORY]** section. First, we add `()` to `trjfile1` due to multiple trajectory files. Second, `COOR+BOX` is not necessary because the box size information is already written to each trajectory file. Below there is the **control file** for RMSD calculation.

```
[INPUT]
psffile = ../1_setup/ionize.psf               # protein structure file
reffile = ../1_setup/ionize.pdb               # PDB file

[OUTPUT]
rmsfile = run.rms                             # RMSD file

[TRAJECTORY]
trjfile1        = ../5_production/run().dcd   # trajectory file
md_step1        = 500000                      # number of MD steps
mdout_period1   = 500                         # MD output period
ana_period1     = 500                         # analysis period

[SELECTION]
group1          = backbone and an:CA # selection group 001

[FITTING]
fitting_method  = TR+ROT                      # method
fitting_atom    = 001                         # atom group

[OPTION]
check_only      = NO                          # (YES/NO)
```

Finally, with this we obtain the same RMSD values as in *Tutorial 1: Building and Simulating BPTI in Water*.

BIBLIOGRAPHY

[1] D. A. Case, T. A. Darden, T. E. Cheatham, III, C. L. Simmerling, J. Wang, R. E. Duke, R. Luo, R. C. Walker, W. Zhang, K. M. Merz, B. Roberts, S. Hayik, A. Roitberg, G. Seabra, J. Swails, A. W. Goetz, I. Kolossváry, K. F. Wong, F. Paesani, J. Vanicek, R. M. Wolf, J. Liu, X. Wu, S. R. Brozell, T. Steinbrecher, H. Gohlke, Q. Cai, X. Ye, J. Wang, M.-J. Hsieh, G. Cui, D. R. Roe, D. H. Mathews, M. G. Seetin, R. Salomon-Ferrer, C. Sagui, V. Babin, T. Luchko, S. Gusarov, A. Kovalenko, and P. A. Kollman. AMBER 12. University of California, San Francisco, 2012.

[2] B. R. Brooks, C. L. Brooks, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus. CHARMM: The biomolecular simulation program. *J. Comput. Chem.*, 30(10):1545–1614, 2009. URL: http://dx.doi.org/10.1002/jcc.21287[1], doi:10.1002/jcc.21287[2].

[3] S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel, B. Hess, and E. Lindahl. GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, 29(7):845–854, 2013.

[4] K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, J. K. Salmon, Y. Shan, and D. E. Shaw. Scalable algorithms for molecular dynamics simulations on commodity clusters. In *SC 2006 Conference, Proceedings of the ACM/IEEE*, 11–17. IEEE, 2006.

[5] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, and K. Schulten. Scalable molecular dynamics with NAMD. *J. Comput. Chem.*, 26(16):1781–1802, 2005. URL: http://dx.doi.org/10.1002/jcc.20289[3], doi:10.1002/jcc.20289[4].

[6] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. J. Woods. The Amber biomolecular simulation programs. *J. Comput. Chem.*, 26(16):1668–1688, 2005.

[7] A. D. MacKerell, D. Bashford, M. Bellott, R. L. Dunbrack, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, and M. Karplus. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Phys. Chem. B*, 102(18):3586–3616, 1998.

[1] http://dx.doi.org/10.1002/jcc.21287
[2] http://dx.doi.org/10.1002/jcc.21287
[3] http://dx.doi.org/10.1002/jcc.20289
[4] http://dx.doi.org/10.1002/jcc.20289

[8] A. D. MacKerell, M. Feig, and C. L. Brooks. Improved treatment of the protein backbone in empirical force fields. *J. Am. Chem. Soc.*, 126(3):698–699, 2004.

[9] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives. Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. *J. Am. Chem. Soc.*, 118(45):11225–11236, 1996.

[10] C. Oostenbrink, A. Villa, A. E. Mark, and W. F. Van Gunsteren. A biomolecular force field based on the free enthalpy of hydration and solvation: The GROMOS force-field parameter sets 53A5 and 53A6. *J. Comput. Chem.*, 25(13):1656–1676, 2004.

[11] J. Jung, T. Mori, and Y. Sugita. Efficient lookup table using a linear function of inverse distance squared. *J. Comput. Chem.*, 34(28):2412–2420, 2013.

[12] J. Jung, T. Mori, and Y. Sugita. Midpoint cell method for hybrid (MPI+OpenMP) parallelization of molecular dynamics simulations. *J. Comput. Chem.*, 35(14):1064–1072, 2014.

[13] Open MPI. http://www.open-mpi.org/.

[14] CHARMM. http://www.charmm.org/.

[15] psfgen. http://www.ks.uiuc.edu/Research/vmd/plugins/psfgen/ug.pdf.

[16] NAMD. http://www.ks.uiuc.edu/Research/namd/.

[17] VMD. http://www.ks.uiuc.edu/Research/vmd/.

[18] Y. Sugita and Y. Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chem. Phys. Lett.*, 314(1–2):141–151, 1999.

[19] Y. Sugita, A. Kitao, and Y. Okamoto. Multidimensional replica-exchange method for free-energy calculations. *J. Chem. Phys.*, 113(15):6042–6051, 2000.

[20] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein. Comparison of Simple Potential Functions for Simulating Liquid Water. *J. Chem. Phys.*, 79(2):926–935, 1983.

[21] N. Foloppe and A. D. Mackerell. All-atom empirical force field for nucleic acids: I. parameter optimization based on small molecule and condensed phase macromolecular target data. *J. Comput. Chem.*, 21(2):86–104, 2000.

[22] A. D. Mackerell and N. K. Banavali. All-atom empirical force field for nucleic acids: II. application to molecular dynamics simulations of dna and rna in solution. *J. Comput. Chem.*, 21(2):105–120, 2000.

[23] S. D. Feller, D. X. Yin, R. W. Pastor, and A. D. Mackerell. Molecular dynamics simulation of unsaturated lipid bilayers at low hydration: parameterization and comparison with diffraction studies. *Biophys. J.*, 73(5):2269–2279, 1997.

[24] J. B. Klauda, R. M. Venable, J. A. Freites, J. W. O'Connor, D. J. Tobias, C. Mondragon-Ramirez, I. Vorobyov, and R. W. Pastor. Update of the charmm all-atom additive force field for lipids: validation on six lipid types. *J. Phys. Chem. B*, 114(23):7830–7843, 2010.

[25] R. B. Best, X. Zhu, J. Shim, P. E. M. Lopes, J. Mittal, M. Feig, and A. D. MacKerell. Optimization of the additive CHARMM all-atom protein force field targeting improved sampling of the backbone \phi , \psi and side-Chain \chi 1 and \chi 2 dihedral angles. *J. Chem. Theo. Comput.*, 8(9):3257–3273, 2012.

[26] J. Huang and A. D. MacKerell. CHARMM36 all-atom additive protein force field: Validation based on comparison to NMR data. *J. Comput. Chem.*, 34(25):2135–2145, 2013.

[27] J. Karanicolas and C. L. Brooks, III. The origins of asymmetry in the folding transition states of protein L and protein G. *Protein Sci.*, 11(10):2351–2361, 2002.

[28] J. Karanicolas and C. L. Brooks III. Improved Go-like models demonstrate the robustness of protein folding mechanisms towards non-native interactions. *J. Mol. Biol.*, 334(2):309–325, 2003.

[29] N. Go. Theoretical studies of protein folding. *Annu. Rev. Biophys. Bioeng.*, 12:183–210, 1983.

[30] P. J. Steinbach and B. R. Brooks. New Spherical-Cutoff Methods for Long-Range Forces in Macromolecular Simulation. *J. Comput. Chem.*, 15(7):667–683, 1994.

[31] L. Verlet. Computer Experiments on Classical Fluids .I. Thermodynamical Properties of Lennard-Jones Molecules. *Phys. Rev.*, 159(1):98–103, 1967.

[32] T. Darden, D. York, and L. Pedersen. Particle mesh Ewald: An Nlog(N) method for Ewald sums in large systems. *J. Chem. Phys.*, 98(12):10089–10092, 1993.

[33] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen. A smooth particle mesh Ewald method. *J. Chem. Phys.*, 103(19):8577–8593, 1995.

[34] D. Takahashi. FFTE: A Fast Fourier Transform Package. http://www.ffte.jp/.

[35] FFTW. http://www.fftw.org/.

[36] L. Nilsson. Efficient Table Lookup Without Inverse Square Roots for Calculation of Pair Wise Atomic Interactions in Classical Simulations. *J. Comput. Chem.*, 30(9):1490–1498, 2009.

[37] J. P. Ryckaert, G. Ciccotti, and H. J. C. Berendsen. Numerical-Integration of Cartesian Equations of Motion of a System with Constraints - Molecular-Dynamics of N-Alkanes. *J. Comput. Chem.*, 23(3):327–341, 1977.

[38] H. C. Andersen. Rattle - a Velocity Version of the Shake Algorithm for Molecular-Dynamics Calculations. *J. Comput. Chem.*, 52(1):24–34, 1983.

[39] S. Miyamoto and P. A. Kollman. Settle - an Analytical Version of the Shake and Rattle Algorithm for Rigid Water Models. *J. Comput. Chem.*, 13(8):952–962, 1992.

[40] S. A. Adelman and J. D. Doll. Generalized Langevin Equation Approach for Atom-Solid-Surface Scattering - General Formulation for Classical Scattering Off Harmonic Solids. *J. Chem. Phys.*, 64(6):2375–2388, 1976.

[41] D. Quigley and M. I. J. Probert. Langevin dynamics in constant pressure extended systems. *J. Chem. Phys.*, 120(24):11432–11441, 2004.

[42] A. Mitsutake, Y. Sugita, and Y. Okamoto. Generalized-ensemble algorithms for molecular simulations of biopolymers. *Biopolymers*, 60(2):96–123, 2001.

[43] J. A. McCammon, B. R. Gelin, and M. Karplus. Dynamics of folded proteins.. *Nature*, 267(5612):585–590, 1977.

[44] D. E. Shaw, P. Maragakis, K. Lindorff-Larsen, S. Piana, R. O. Dror, M. P. Eastwood, J. A. Bank, J. M. Jumper, J. K. Salmon, Y. Shan, and W. Wriggers. Atomic-level characterization of the structural dynamics of proteins.. *Science*, 330(6002):341–346, 2010.

[45] gnuplot. http://www.gnuplot.info/.

[46] RCSB Protein Data Bank. http://www.rcsb.org/.

[47] solvate plugin. http://www.ks.uiuc.edu/Research/vmd/plugins/solvate/.

[48] autoionize plugin. http://www.ks.uiuc.edu/Research/vmd/plugins/autoionize/.

[49] MMTSB web service. http://mmtsb.org/webservices/gomodel.html.

[50] Temperature generator for REMD-simulations. http://folding.bmc.uu.se/remd/.