GENESIS User Guide

Release 1.3.0

RIKEN

Jun 14, 2018

GENESIS 1.3.0

Project Leader: Yuji Sugita (RIKEN)

Main Developers for GENESIS 1.1.x, 1.2.x, 1.3.x: Chigusa Kobayashi (RIKEN), Jaewoon Jung (RIKEN), Yasuhiro Matsunaga (RIKEN), Takaharu Mori (RIKEN), Tadashi Ando (RIKEN), Koichi Tamura (RIKEN), Motoshi Kamiya (RIKEN)

Main Developers for GENESIS 1.0: Jaewoon Jung (RIKEN), Takaharu Mori (RIKEN), Chigusa Kobayashi (RIKEN), Yasuhiro Matsunaga (RIKEN), Takao Yoda (Nagahama Institute of Bio-Science and Technology), Michael Feig (Michigan state university)

Contributors: Takashi Imai (RIKEN), Ryuhei Harada (RIKEN), Yasuaki Komuro (RIKEN), Raimondas Galvelis (RIKEN), Yu Isseki (RIKEN), Kiyoshi Yagi (RIKEN), Daisuke Matsuoka (RIKEN), Donatas Surblys (RIKEN), Suyong RE (RIKEN), Wataru Nishima (RIKEN), Naoyuki Miyashita (RIKEN),

Acknowledgments: Norio Takase (Isogo Soft), Akira Naruse (NVIDIA), Yukihiko Hirano (NVIDIA Japan) Hikaru Inoue (Fujitsu Ltd.) Tomoyuki Noda (Fujitsu Ltd.)

Citation Information

C. Kobayashi, J. Jung, Y. Matsunaga, T. Mori, T. Ando, K. Tamura, M. Kamiya, and Y. Sugita, "GENESIS 1.1: A hybrid-parallel molecular dynamics simulator with enhanced sampling algorithms on multiple computational platforms", J. Comput. Chem. 38, 2193-2206 (2017)

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 5, 310-323 (2015).

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.

SIMD-oriented Fast Mersenne Twister (SFMT)

SFMT is a new variant of Mersenne Twister (MT) introduced by Mutsuo Saito and Makoto Matsumoto in 2006. The algorithm was reported at MCQMC 2006. The routine is distributed under the New BSD License.

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.

FFTE: A Fast Fourier Transform Package

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

> Copyright ©2000-2004, 2008-2011 Daisuke Takahashi (Tsukuba University).
>
> You may use, copy, modify this code for any purpose (include commercial use) and without fee. You may distribute this ORIGINAL package.

Complementary error function: erfc04

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

> Copyright ©1993 Sun Microsystems, Inc.
>
> Developed at SunSoft, a Sun Microsystems, Inc. business. Permission to use, copy, modify, and distribute this software is freely granted, provided that this notice is preserved (see at_energy_pme.fpp, and sp_energy_pme.fpp).

Note:

In computational neural science, there is another "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, 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 hundreds of CPUs or CPU cores can be used efficiently for a single MD simulation of a relatively large biomolecular system, typically composed of several hundred thousands of atoms. In recent years, the number of available CPUs or CPU cores is rapidly increasing. The implmentation of highly efficient parallel schemes is therefore required in modern MD simulation programs. Accelerators such as GPGPU (General-Purpose computing on Graphics Processing Units) also become popular, and thus their utilization is also desired. Actually, many MD program packages support various accelerators.

Our major motivation is to develop MD simulation software with a scalable performance on such modern supercomputers. For this purpose, we have developed the software from scratch, introducing the hybrid (MPI + OpenMP) parallelism, several new parallel algorithms *[11] [12]*, and GPGPU support by NVIDIA CUDA technology. Another motivation is to develop a MD simulation code, which can be easily understood and modified for methodological developments. These two policies, high parallel performance and simplicity, usually conflict each other in computer software. To avoid the conflict, we developed two MD programs, namely **SPDYN** (*Sp*atial decomposition *dyn*amics) and **ATDYN** (*At*omic decomposition *dyn*amics).

These two MD codes share almost the same data structures, subroutines, and modules, but differ in their parallelization schemes. In **SPDYN**, the spatial decomposition scheme is implemented with new parallel algorithms *[11] [12]* and GPGPU support. In **ATDYN**, the atomic decomposition scheme is introduced for simplicity.

The performance of **ATDYN** is not comparable to **SPDYN** due to the simple parallelization scheme. However, **ATDYN** is easier to modify for development of new algorithms or novel molecular models. We hope that users develop new methodologies in **ATDYN** at first and, eventually, port them to **SPDYN** for the better performance. As we maintain consistency between the source codes of **ATDYN** and **SPDYN**, switching from **ATDYN** to **SPDYN** is not quite hard.

Other features in **GENESIS** are listed below:

- Not only atomistic molecular force field (CHARMM, AMBER) but also some coarse-grained models are available in **ATDYN**.
- For extremely large biomolecular systems (more than 10 million atoms), parallel input/output (I/O) scheme is implemented.
- **GENESIS** is optimized for "K computer" (developed by RIKEN and Fujitsu company), but it is also available on Intel-based supercomputers and 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 users to use/modify **GENESIS** and redistribute the modified version under the same license.

This manual consists of 14 chapters including how to get started and explanation of each keyword in control files. Tutorials for standard MD simulations, REMD simulations, and some analyses are available online (http://www.r-ccs.riken.jp/labs/cbrt/). 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.

Comparing to other MD software, e.g. AMBER, CHARMM, or NAMD, **GENESIS** is a very young MD simulation program. 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 might be defects or bugs in **GENESIS**. Since we cannot bear any responsibility for the simulation results produced by **GENESIS**, we strongly recommend users to check their results carefully.

The **GENESIS** development team has a rich plan for future development of methodology and molecular models. We would like to make **GENESIS** one of the most powerful and feasible MD software packages, contributing to computational chemistry and biophysics. Computational studies in life science is still at a very early stage (like 'GENESIS') compared to established experimental researches. We hope that **GENESIS** pushes forward the computational science and contribute to bio-tech and medical applications in the 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 such as **trj_analysis**, **crd_convert**, **pcrd_convert**, **remd_convert**, are used to post-process the trajectories produced by simulators. **prst_setup** generates **GENESIS** original parallel restart I/O data for the huge system from conventional input data.

A description of each program is given in the next chapter (*Available Programs*). Users could find detailed usage (including references for input parameters) in the online tutorials (http://www.r-ccs.riken.jp/labs/cbrt/tutorial/). This chapter is devoted to help new users playing with **GENESIS**. The first half of this chapter covers compilation and installation of **GENESIS**. In the second half, we give the users a general idea of how to use **GENESIS**.

2.1 How to obtain GENESIS

GENESIS source code is distributed under the GNU General Public License version 2. Everyone can download **GENESIS** freely at GENESIS download page (http://www.r-ccs.riken.jp/labs/cbrt/download/), where a dataset for test is also available.

If you have any problems in downloading **GENESIS**, please report to the forum (http://www.r-ccs.riken.jp/labs/cbrt/genesis-forum/). Note that you have to register for the forum to make a comment.

2.2 How to install GENESIS

2.2.1 Requirements

The source code of **GENESIS** is written in Fortran 90/95/2003 and organized in *modules*. The preprocessing directives (such as `#ifdef`) are used to adapt the source code for different architectures and options. Fortran compiler and C preprocessor are the minimum requirements 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 process. Whereas OpenMP is natively supported by most of modern Fortran compilers, you may need to install MPI library in your environment.

Although **GENESIS** is tested on various platforms, at present, limited combinations of CPU and compiler from Intel and Fujitsu are validated. Thus, the combination of Intel CPU and compiler, or SPARC

CPU and Fujitsu compiler is recommended. As for the MPI library, we also recommend OpenMPI *[13]* that have throughly tested with **GENESIS**.

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 is required; some features do not work)
- C Preprocessor
  - `fpp` supplied with Intel compiler (Recommended)
  - Fujitsu compiler `frtpx` (Recommended; it also supports preprocessing)
  - GCC preprocessor `cpp`
- MPI implementation
  - OpenMPI (Recommended)
  - Fujitsu MPI (Recommended)
  - Intel MPI
- Operating system
  - Linux (Recommended)
  - Mac OSX (Not recommended, but may work)
- GPGPU and CUDA
  - NVIDIA GPU cards which support Compute Capability (CC) 3.5 (Kepler) or higher are required (CC 3.0 cards such as GTX 680 are not supported)
  - NVIDIA Tesla K20, K40 (Tested; GTX cards may also work)
  - CUDA 6.5, 7.5, 8.0 (Tested)

2.2.2 Installation

First, extract the archive of the source code (`genesis-1.3.0.tar.bz2`).

```
### untar the package file and change the working directory
$ tar xvfj genesis-1.3.0.tar.bz2
$ cd genesis/
$ ls -lF
total 32
-rw-rw-r-- 1 user  staff      0 Dec  8 17:06 AUTHORS
-rw-rw-r-- 1 user  staff  17988 Dec  8 17:06 COPYING       # License
-rw-rw-r-- 1 user  staff      0 Dec  8 17:06 ChangeLog
-rw-rw-r-- 1 user  staff      0 Dec  8 17:06 INSTALL
-rw-rw-r-- 1 user  staff     49 Dec  8 17:06 Makefile.am    #
-rw-rw-r-- 1 user  staff  21666 Dec  8 17:06 Makefile.in
-rw-rw-r-- 1 user  staff      0 Dec  8 17:06 NEWS
-rw-rw-r-- 1 user  staff   8389 Dec  8 17:06 README        # README file
```

```
-rw-rw-r-- 1 user  staff  35662 Dec  8 17:06 aclocal.m4
-rwxrwxr-x 1 user  staff   7333 Dec  8 17:06 compile*
-rwxrwxr-x 1 user  staff 203671 Dec  8 17:06 configure*
-rw-rw-r-- 1 user  staff  26179 Dec  8 17:06 configure.ac
-rwxrwxr-x 1 user  staff  23566 Dec  8 17:06 depcomp*
-rwxrwxr-x 1 user  staff  14675 Dec  8 17:06 install-sh*
-rwxrwxr-x 1 user  staff   6872 Dec  8 17:06 missing*
drwxrwxr-x 1 user  staff    131 Dec  8 17:17 src/         # Source codes
```

The source tree has been changed in version 1.3.0.

GENESIS uses GNU autotools build system. Run `./configure` script to create Makefiles for your system. The `./configure` script tries to detect all of the requirements needed to compile **GENESIS**:

```
### (version >= 1.2.1) Run ./configure in the top directory
### if you wanna use GPU, please see the example below
$ ./configure
```

It may take a while for `./configure` script to complete. While running, it prints messages telling what are checked. It should be noted that this script automatically searches Fujitsu compilers (mpifrtpx, mpifrt) and mpif90 for Fortran, and Fujitsu compilers (mpifccpx, mpifcc) and mpicc for C. If you want to use other compilers (such as mpiifort and mpiicc), please specify them mannually via environmental variables. (See examples below.)

If you need to add compilation options and/or library paths, you can use the following options (you can see the full list by `./configure --help`):

options	meaning
–prefix=PREFIX	install architecture-independent files in `PREFIX`
–exec-prefix=EPREFIX	install architecture-dependent files in `EPREFIX`
–program-prefix=PREFIX	prepend `PREFIX` to installed program names
–program-suffix=SUFFIX	append `SUFFIX` to installed program names
–enable-debug=LEVEL	set Debug level (from 0 to 4) (`--enable-debug` is the same as `--enable-debug=1`)
–enable-single	single precision real numbers are used for calculation
–enable-gpu	GPU is used for real space non-bonded interaction
–disable-openmp	disable OpenMP parallelization
–with-cuda=PATH	CUDA is assigned for compiling gpu source code
–without-lapack	do not use LAPACK
–disable-parallel_IO	disable parallel IO (skip build of prst_setup)
–host=host	disable compiler check (**Required** in cross compiler system)

Here, the following options are available only in SPDYN: `--enable-single`, `--enable-gpu`, `--with-cuda`, and `--disable-parallel_IO`. In Fujitsu compilers at K-computer (`--host=k`), *parallel_IO* is practically not available due to the file size as well as job time limitations and thus is disabled automatically.

In single precision, *parallel_IO* and analysis tools are not installed.

There are some environmental variables which can be used in the configuration.

env variable name	meaning
FC	Fortran compiler command
FCFLAGS	Fortran compiler flags
CC	C compiler command
CFLAGS	C compiler flags
LAPACK_PATH	LAPACK library path (linker flag, e.g. -L<lapack dir>)
LAPACK_LIBS	LAPACK library files (libraries, e.g. -l<lapack library name>)
CUDA_LIB_PATH	CUDA library path (linker flag, e.g. -L<cuda lib>

Example of configurations:

```
### example for non-cross-compilation system
$ ./configure

### example for K computer
$ ./configure --host=k

### example for HOKUSAI Greatwave (FX100) in RIKEN
$ module load sparc
$ ./configure --host=k

### example for mpiifort and mpiicc using environmental variables
### you may modify compilation flags using FCFLAGS and CFLAGS
$ FC=mpiifort CC=mpiicc ./configure

### example for GPGPU with CUDA on PC clusters and Workstations
### if you wanna specify CUDA manually, please use --with-cuda=/path/to
$ ./configure --enable-single --enable-gpu

### example of debugging mode with higher level (time consuming)
### enable-debug=LEVEL: 0 = no debugging (default)
###                     1 = without intensive optimization
###                     2 = LEVEL=1 & with debug information (-g) & -DDEBUG
###                     3 = LEVEL=2 & memory check (if possible)
###                     4 = LEVEL=3 & full check (intel compiler only)
### if LEVEL is not given explicitly, LEVEL is 1.
$ ./configure --enable-debug=3

### 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
```

From **GENESIS 1.2.0**, parallel compilation is officially supported. -j option can be added to the make command if you can use multiple CPU cores:

```
$ make -j8 install
```

`make install` compiles all of the **GENESIS** programs (the simulators and the analysis tools). The compiled binary files are copied to `bin/` directory (default; you can change installation path in

`configure`). If the compilations are successfully finished, the following binary files has to be in your `bin/`:

```
$ ls -lF bin/
total 106336
-rwxr-xr-x 1 user staff  8139755 Jan 26 09:44 atdyn
-rwxr-xr-x 1 user staff 11624014 Jan 26 09:44 spdyn
-rwxr-xr-x 1 user staff 13267683 Jan 26 09:44 prst_setup
-rwxr-xr-x 1 user staff  7267743 Jan 26 09:44 crd_convert
-rwxr-xr-x 1 user staff  7305386 Jan 26 09:44 pcrd_convert
-rwxr-xr-x 1 user staff  7786868 Jan 26 09:44 remd_convert
-rwxr-xr-x 1 user staff  2359193 Jan 26 09:44 rst_convert
-rwxr-xr-x 1 user staff  7268648 Jan 26 09:44 avecrd_analysis
-rwxr-xr-x 1 user staff  7283268 Jan 26 09:44 flccrd_analysis
-rwxr-xr-x 1 user staff  6466979 Jan 26 09:44 eigmat_analysis
-rwxr-xr-x 1 user staff  7277618 Jan 26 09:44 prjcrd_analysis
-rwxr-xr-x 1 user staff  2732713 Jan 26 09:44 trj_analysis
-rwxr-xr-x 1 user staff  2906475 Jan 26 09:44 wham_analysis
-rwxr-xr-x 1 user staff  2927918 Jan 26 09:44 mbar_analysis
-rwxr-xr-x 1 user staff  2649040 Jan 26 09:44 qval_analysis
-rwxr-xr-x 1 user staff  7263330 Jan 26 09:44 rmsd_analysis
```

Note: **SPDYN** can use CUDA-enabled NVIDIA GPUs as accelerators for the computation of nonbonding interactions. GPUs which support CC 3.5 (Kepler) or higher are required. GPUs of early Kepler (CC 3.0) or even earlier generations cannot be used due to the hardware restrictions.

2.2.3 Regression Tests

A set of tests are prepared for **ATDYN**, **SPDYN** and **prst_setup** (parallel I/O) to check if these programs work correctly. The tarball of the test is also availale online (http://www.r-ccs.riken.jp/labs/cbrt/download/).

```
### untar the package file and change the working directory
$ tar xvfj tests-1.3.0.tar.bz2
$ cd tests-1.3.0/regression_test
$ ls -lF
total 120
drwxr-xr-x 12 user staff  4096 Dec 22 17:28 build/
-rwxr-xr-x  1 user staff 29532 Dec 22 17:28 charmm.py
-rwxr-xr-x  1 user staff 10612 Dec 22 17:28 genesis.py
drwxr-xr-x  3 user staff  4096 Dec 22 17:28 param/
-rwxr-xr-x  1 user staff  9953 Dec 22 17:28 test.py
drwxr-xr-x  7 user staff  4096 Dec 22 17:28 test_atdyn/
drwxr-xr-x  9 user staff  4096 Dec 22 17:28 test_common/
-rwxr-xr-x  1 user staff  5695 Dec 22 17:28 test_nonstrict.py
drwxr-xr-x  7 user staff  4096 Dec 22 17:28 test_parallel_IO/
drwxr-xr-x  6 user staff  4096 Dec 22 17:28 test_remd/
-rwxr-xr-x  1 user staff  3202 Dec 22 17:28 test_remd.csh
-rwxr-xr-x  1 user staff  5655 Dec 22 17:28 test_remd.py
-rwxr-xr-x  1 user staff  5796 Dec 22 17:28 test_rpath.py
drwxr-xr-x  3 user staff  4096 Dec 22 17:28 test_rpath_atdyn/
drwxr-xr-x  3 user staff  4096 Dec 22 17:28 test_rpath_spdyn/
```

```
drwxr-xr-x  5 user staff  4096 Dec 22 17:28 test_spdyn/
```

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

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

Here, [genesis_command] is a command line for executing **ATDYN** or **SPDYN** (default: `mpirun -np 8 atdyn`). Note that the OpenMP thread number has to be specified explicitly before running. *parallel_io* needs to be appended to check the parallel I/O facility, and *gpu* must be appended if you complied **SPDYN** with `--enable-gpu`.

```
### run atdyn test (single thread)
$ export OMP_NUM_THREADS=1
$ ./test.py "mpirun -np 8 /path/to/atdyn"

### run spdyn test (single thread)
$ export OMP_NUM_THREADS=1
$ ./test.py "mpirun -np 8 /path/to/spdyn"

### run GPU enabled spdyn test (single thread)
$ export OMP_NUM_THREADS~1
$ ./test.py "mpirun -np 8 /path/to/spdyn" gpu

### run parallel_io test (single thread)
$ export OMP_NUM_THREADS=1
$ ./test.py "mpirun -np 8 /path/to/spdyn" parallel_io

### run REMD test with atdyn (single thread)
$ export OMP_NUM_THREADS=1
$ ./test_remd.py "mpirun -np 8 /path/to/atdyn"

### run REMD test with spdyn (single thread)
$ export OMP_NUM_THREADS=1
$ ./test_remd.py "mpirun -np 8 /path/to/spdyn"
```

You can use `mpiexec` or other laucher commands instead. For example,

```
$ ./test.py "mpiexec -n 8 /path/to/spdyn"
```

will work correctly if properly configured. But, the name of **GENESIS** simulator must be *"atdyn"* or *"spdyn"*. If you change the executable names, the regression test fails to run.

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 might fail.

Note: **prst_setup** is disabled automatically when compiled with the Fujitsu compiler. There is no problem in running **SPDYN** using parallel restart files generated by **prst_setup** complied with other compilers.

2.3 Basic Usage

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 in the following way:

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

Usually, you should specify OpenMP thread number explicitly before running programs (thread number might be 1, 2, 3, ...). Note that the appropriate OpenMP/MPI ratio depends on the simulation system size, number of cpu cores, and hardware (such as CPUs, GPUs, and network).

```
$ export OMP_NUM_THREADS=1
$ mpirun -np 8 /path/to/spdyn ctrl.inp
```

GENESIS program generates a template of the control file when executed with `-h ctrl [template_name]` option. A list of available *template_name* can be obtained by running with `-h` option only. For example, **SPDYN** prints the template names for `md` (molecular dynamics), `min` (minimization), `remd` (replica exchange molecular dynamics), and `rpath` (replica path sampling):

```
$ spdyn -h

# normal usage
% ./spdyn INP

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

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

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

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

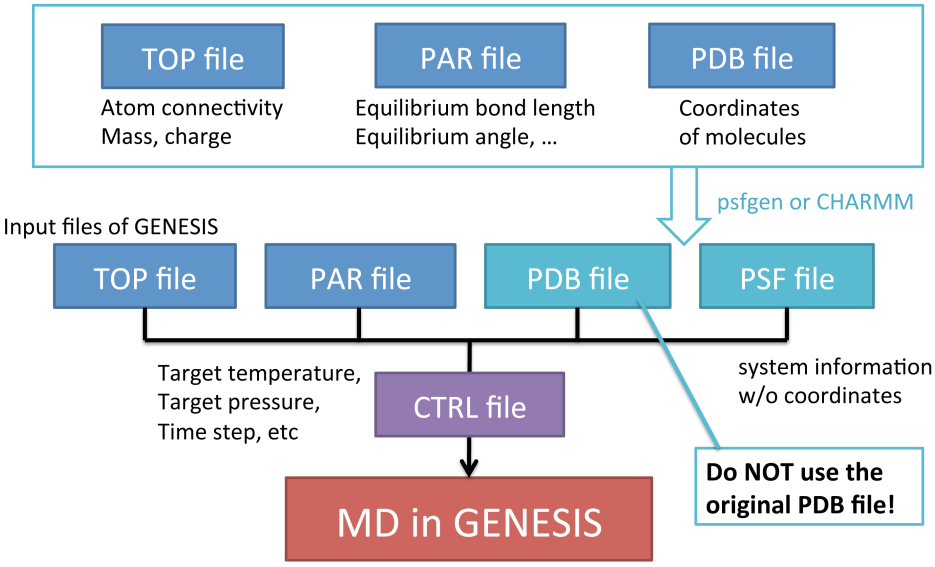
If you are interested in all available options including detailed parameters for advanced users, please use `ctrl_all` instead of `ctrl`. For example, running with `-h ctrl_all min` will print all available options for minimization.

2.3.1 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. Currently, **GENESIS** supports CHARMM/NAMD, AMBER, and GROMACS styles (with some limitations, though).

The coordinates can be given by various formats such as *PDB*, CHARMM *CRD*, AMBER restart, or **GENESIS** restart files (*pdbfile*, *crdfile*, *ambcrdfile*, or *rstfile*, respectively; see *Input and Output files* for details). If you employ CHARMM force field, you may need to specify CHARMM parameter, topology, and/or stream files (*parfile*, *topfile*, and *strfile*, respectively) along with the PSF file (*psffile*). Details about file specifications can be found at *Input and Output files* section.

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]*, psfgen *[15]* (supplied with NAMD *[16]* and VMD *[17]*), AMBER LEaP *[18]*, or SMOG *[19]*. Setup procedures are also demonstrated in the online tutorials (http://www.r-ccs.riken.jp/labs/cbrt/tutorial/). An example information flow of input files for the CHARMM/NAMD style in **GENESIS** is summarized in a chart below.



2.3.2 Control File

The input files and control parameters of **GENESIS** have to be specified in a *control file*. The control file consists of several *sections* (ex. **[INPUT]**, **[OUTPUT]**, **[ENERGY]**), where each section contains a set of *keywords*.

Note: Each parameter string shall be less than *MaxLine* (**Default : 1000**) characters. *MaxLine* is defined in "src/lib/string.fpp". If a statement is too long, you can use multiple lines for a single parameter by appending a line continuation character, backslash "", to the end of the line. See example in the *Input and Output files* section for this issue.

For example, important sections and keywords (when CHARMM style input and forcefield are employed) are shown 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 / AMBER etc.*)
  - **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 / VRES*)
  - **nsteps**: number of MD steps (*default: 100*)
  - **timestep**: time interval of MD step (*default: 0.001*)
  - **nbupdate_period**: frequency (in MD steps) of a non-bonded pair-list updates (*default: 10*)
- **[MINIMIZE]** section: minimization (see *Minimize*)
  - **nsteps**: number of minimization steps (*default: 100*)
  - **nbupdate_period**: frequency (in minimization steps) of non-bonded pair-list updates (*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 / NPAT / NPgT*)
  - **tpcontrol**: type of thermostat and barostat (*NO / BERENDSEN / LANGEVIN / BUSSI*)
  - **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 (*PBC / NOBC*)

2.3.3 Minimization

A sample control file for the energy minimization is shown below. **[INPUT]** section contains the input file names, and the parameters of **[ENERGY]** specify energy and force evaluation method. **[MINIMIZE]** section defines the number of steps and frequencies of output and pairlist update of this minimization run.

```
[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 etc.]
electrostatic   = PME                         # [CUTOFF,PME]
switchdist      = 10.0                        # switch distance
cutoffdist      = 12.0                        # cutoff distance
pairlistdist    = 13.5                        # pairlist distance
contact_check   = YES                         # activate checker

[MINIMIZE]
nsteps          = 100                         # number of steps
eneout_period   = 10                          # energy output freq
crdout_period   = 10                          # coordinates output freq
rstout_period   = 100                         # restart output freq
nbpdate_period  = 10                          # pairlist update freq

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

2.3.4 Molecular Dynamics

A sample control file for the molecular dynamics simulation is shown below. In this case, instead of **[MINIMIZE]** section, **[DYNAMICS]** section defines the integrator and various frequencies. **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 etc.]
electrostatic   = PME                       # [CUTOFF,PME]
switchdist      = 8.0                       # switch distance
cutoffdist      = 12.0                      # cutoff distance
pairlistdist    = 13.5                      # 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 freq
crdout_period   = 10                        # coordinates output freq
rstout_period   = 1000                      # restart output freq
nbupdate_period = 10                        # pairlist update freq

[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 trajectory and restart files.

3.1 Simulators

GENESIS has two simulators with different decomposition schemes, but Input/Output files adhere to the common formats (except for the *parallel I/O* scheme). Basic functionalities are also common in **ATDYN** and **SPDYN**; molecular dynamics, energy minimization, and replica exchange molecular dynamics (REMD) *[20] [21]* simulations are available in both of the simulators. However, some of simulation options are still specific to **ATDYN** or **SPDYN**.

ATDYN (ATomic decomposition DYNamics simulator)

> 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.). Go-models are available only in this simulator.

SPDYN (SPatial decomposition DYNamics simulator)

> The simulator uses the spatial decomposition scheme with our new algorithms, including the parallel I/O scheme, for a better parallel scaling performance. The simulator is more complex than **ATDYN** and designed for high performance and good scalability on massively parallel computers. Moreover, NVIDIA GPU can be used for the acceleration of nonbonding interaction energy/force computations.

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 briefly introduced in the online tutorials (http://www.r-ccs.riken.jp/labs/cbrt/tutorial/).

trj_analysis

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

avecrd_analysis

A utility to compute the average structure of a molecule by fitting the structure to a putative average structure repeatedly.

flccrd_analysis

A utility to calculate a variance-covariance matrix from trajectory file.

eigmat_analysis

A utility for eigenvalue decomposition of a variance-covariance matrix. This is used for the principal component analysis.

prjcrd_analysis

A utility for projecting trajectory to principal component axes.

wham_analysis

A utility to perform the analysis using Weighted Histogram Analysis Method (WHAM).

mbar_analysis

A utility to perform the analysis using Multistate Bennett Acceptance Ratio (MBAR) method.

qval_analysis

A utility to compute Q-value (the fraction of native contacts) from a trajectory file.

crd_convert

A 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 condition by reading the original trajectory files and replica id log files (remfile) of all replicas.

rst_convert

Restart file interconverter between NAMD and GENESIS.

rst_upgrade

GENESIS restart file converter from the old format (for GENESIS < 1.1.0) to the new format (for GENESIS >= 1.1.0).

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 the online tutorial (http://www.r-ccs.riken.jp/labs/cbrt/tutorial/advanced_md_tutorials/tutorial-3-1/).

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 can be found in the online tutorial.

MD runs with parallel IO restart files are available on K-computer, although compilation of *prst_setup* utility is disabled. If you want to perform MD runs with parallel IO in Fujitsu supercomputers, you should run *prst_setup* first on the other computers such as intel CPU clusters to prepare input files.

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

ATDYN and **SPDYN** use the following input/output files. The input and output files are specified in **[INPUT]** and **[OUTPUT]** sections, respectively.

4.1 Input files

ATDYN and **SPDYN** requires molecular topology, coordinate, and parameter files describing the simulation system and potential energy functions. **GENESIS** simulators currently support CHARMM, AMBER, and GROMACS styles. Required files depend on the input style. See also **[ENERGY]** section (*Energy*) for the forcefield specification.

Atomic coordinates can be given by PDB (pdbfile), CHARMM CRD (crdfile), AMBER restart (ambcrdfile), GROMACS coordinate (grocrdfile), and **GENESIS** restart (rstfile) files.

If you specify "pdbfile", this file overwrites the system given by "crdfile", "ambcrdfile", and "grocrdfile".

If you specify "rstfile", coordinates (given by "pdbfile" or others), velocities and other properties of the system (such as simulation box size) are overwritten.

Typically, CHARMM style will require *topfile*, *parfile*, *psffile*, *strfile*, and *pdbfile/crdfile*. AMBER style will require *prmtopfile* and *pdbfile/ambcrdfile*. GROMACS style will require *grotopfile* and *pdbfile/grocrdfile*.

---

topfile

(*CHARMM*) CHARMM topology file containing information about atom connectivity of residues and other molecules. For details on the format, see the CHARMM web site *[14]*. Multiple files can be specified in "topfile" (see note below). (**Optional**)

parfile

(*CHARMM*) CHARMM parameter file containing force field parameters, e.g. force constants and equilibrium geometries. Multiple files can be specified in "parfile" (see note below). (**Optional**)

strfile

(*CHARMM*) CHARMM stream file containing additional force field parameters. Multiple files can be specified in "strfile" (see note below). (**Optional**)

psffile

(*CHARMM*) CHARMM/X-PLOR 'psffile' containing information of the system such as atomic masses, charges, and atom connectivities. (**Optional**)

prmtopfile

(*AMBER*) AMBER 'PARM' or 'prmtop' file (AMBER7 or later format) containing information of the system such as atomic masses, charges, and atom connectivities. For details about this format, see the AMBER web site *[18]*. (**Optional**)

grotopfile

(*GROMACS*) Gromacs 'top' file containing information of the system such as atomic masses, charges, atom connectivities. For details about this format, see the Gromacs web site *[22]*. (**Optional**)

pdbfile

Atomic coordinates in PDB (Protein Data Bank) format. If *pdbfile* is given, coordinates in *crdfile*, *grocrdfile*, *ambcrdfile* are discarded. (**Optional**)

crdfile

(*CHARMM*) Atomic coordinates in CHARMM format. If *pdbfile* or *rstfile* is also given, coordinates in *crdfile* are overwritten and dicarded. (**Optional**)

ambcrdfile

(*AMBER*) Atomic coordinates in AMBER coordinate/restart format (ascii). If *pdbfile* or *rstfile* is also given, coordinates in *ambcrdfile* are overwritten and discarded. (**Optional**)

grocrdfile

(*GROMACS*) Atomic coordinates in GROMACS format. If *pdbfile* or *rstfile* is also given, coordinates in *grocrdfile* are overwritten and discarded. (**Optional**)

Velocites and simulation box size in this file are ignored for now.

rstfile

This file contains atomic coordinates, velocities, simulation box size and other simulation variables with the double-precision floating-point number representation in **GENESIS**-original format. This 'rstfile' has to be specified when restarting a simulation. If *rstfile* is given, coordinates in *pdbfile*, *crdfile*, *grocrdfile*, *ambcrdfile*, and simulation box size in the control file are ignored. (**Optional**)

In *REMD* and *String method* runs, restart file for each replica can be specified using special string '{}' (e.g. `rstfile = rst1_{}.rst`), which is a collective description of the restart files for different replica IDs. If '{}' is not involved, the specified restart file is used for all the replicas. See corresponding sections and tutorials for details.

Note that coordinate file such as *pdbfile*, *crdfile*, *ambcrdfile*, or *grocrdfile* has to be specified in the restart run. The coordinate file is still required to define the system even when *rstfile* is given.

GENESIS 1.1.0 or later uses the new file format for the restart file, which is, unfortunately, incompatible with the former one. If you want to use restart files from the former releases, please convert the file using the *rst_upgrade* utility.

reffile

Reference coordinates (PDB file format) for positional restraints and coordinate fitting. This file should contain the same total number of atoms as *pdbfile*, *crdfile*, *ambcrdfile*, or *grocrdfile*. (**Optional**)

In case of *String method*, this file is used only for the reference coordinate of the restraints. Please check *fitfile* below for the fitting reference coordinate in the String method.

groreffile

(*GROMACS*) Reference coordinates ('gro' file format) for positional restraints and coordinate fitting. This file should contain the same total number of atoms as *pdbfile* or *grocrdfile*. (**Optional**)

In case of *String method*, this file is used only for the reference coordinate of the restraints. Please check *fitfile* below for the fitting reference coordinate in the String method.

ambreffile

(*AMBER*) Reference coordinates ('amber crd' file format) for positional restraints and coordinate fitting. This file should contain the same total number of atoms as *pdbfile* or *ambcrdfile*. (**Optional**)

In case of *String method*, this file is used only for the reference coordinate of the restraints. Please check *fitfile* below for the fitting reference coordinate in the String method.

fitfile

(*String method* only; GENESIS 1.1.5 or later) Reference coordinate for structure fitting. This is only used in the String method. For other cases (MD, MIN, or REMD), *reffile*, *groreffile*, or *ambreffile* above is used for reference coordinate of fitting and this *fitfile* is simply ignored. (**Optional**)

modefile

Principal modes used with principal component (PC) restraints. This file contains only single column ascii data. The XYZ values of each atom's mode vector are stored from the low-index modes. (**Optional**)

localresfile (available for **SPDYN** only)

This "localresfile" defines external forces of "local restraint". (**Optional**)

This is the local restraining potentials for the spatial decomposition scheme. These "local restraints" are different from the restraints defined in **[RESTRAINTS]** and are computationally more efficient than the latter. However, in the local restraints, specified atoms has to be in the same cell and the potential energies are added to normal bond, angle, dihedral energies unlike restraints in **[RESTRAINTS]**. If you want to distinguish restraint potential energies from the ordinary energies, use the **[RESTRAINTS]** section instead.

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 simply added to the corresponding energy terms (bond, angle, or dihedral) in the output file.

The file should contain the following information:

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

where indices of "atom" index 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 defined in **[RESTRAINTS]** section (see *Restraints*). Equivalent calculations to `local restraint` can be performed by **[RESTRAINTS]**.

Note: To specify multiple files for *parfile*, *topfile*, and *strfile*, comma-separated lists are available.

```
parfile = par_all36_prot.prm, par_all36_na.prm, par_all36_lipid.prm
```

To specify very long list of filenames, you can use the line continuation character, backslash.

```
parfile = par_all36_prot.prm, \
          par_all36_na.prm, \
          par_all36_lipid.prm
```

4.2 Output files

GENESIS yields trajectory files (coordinates and velocities) in *DCD* format regardless of the forcefield and input styles. **GENESIS** also generates restart file (*rstfile*) during simulations. Output frequencies of each file can be specified in the **[DYNAMICS]** section.

REMD and *String method* simulations expect slightly different file specifications. Please check corresponding sections and tutorials for details.

dcdfile

Name of trajectory file. Coordinates are written in DCD format (also used by X-PLOR and CHARMM). If you need velocities, please check *dcdvelfile* below. (**Required** if `crdout_period > 0`)

This parameter is ignored when `crdout_period <= 0`.

dcdvelfile

Name of the velocity-only trajectory file. Velocities are also written in DCD format. (**Required** if `velout_period > 0`)

This parameter is ignored when `velout_period <= 0`.

rstfile

'rstfile' contains coordinate, velocities, simulation box size and other information about dynamics. An MD simulation can be restarted using the *rstfile* generated by either a previous MD or MIN simulation; a REMD simulation can be restarted from any type of *rstfile* (MIN/MD/REMD). (**Required** if `rstout_period > 0`)

This parameter is ignored when `rstout_period <= 0`.

Note that if rstout_period <= 0, even the final coordinate is not saved. To enable the restart, make sure to set a positive value for rstout_period.

In case of *REMD* and *String method* runs, the filename must contain '{}', which is automatically replaced with a replica ID by the program. See corresponding sections and tutorials for details.

remfile (only for *REMD* simulations)

This file provides parameter ID of each replica at every *exchange_period* steps. This is used as an input file for *remd_convert* utility. In this file name specification, '{}' is replaced with the replica ID. See *REMD* section and tutorials for details. (**Required** if `exchange_period > 0`)

logfile (only for *REMD* and *String method* runs)

This file provides log (such as energies) of each replica. Filename must have '{}' which is replaced with a corresponding replica ID. See *REMD* section and tutorials for details. (**Required** if *REMD* or *String method*)

rpathfile (only for *String method* simulations)

This file provides the trajectory of image coordinates used with the string method (image coordinates are reference values used in restraint functions). Columns correspond to the collective variable and rows are time steps. These data are written at the same timing with *dcdfile* (`crdout_period`) See *String method* section for the string method. (**Required** if *String method*)

CHAPTER

FIVE

ENERGY

In the **[ENERGY]** section, there are several options relating to energy and force evaluations.

5.1 General keywords

The force field consists of a potential energy functions and a set of parameters. In the case of the all-atom force field with explicit water, the potential energy function is given as:

$$
\begin{aligned}
E(r) = & \sum_{\text{bond}} K_b(b-b_0)^2 + \sum_{\text{UB}} K_{ub}(S-S_0)^2 + \sum_{\text{angle}} K_\theta(\theta-\theta_0)^2 \\
& + \sum_{\text{dihedral}} K_\phi(1+\cos(n\phi-\delta)) + \sum_{\text{improper}} K_{\phi_i}(\phi_i-\phi_{i,0})^2 \\
& + \sum_{\text{nonbond}} \epsilon \left[\left(\frac{R_{min,ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{R_{min,ij}}{r_{ij}} \right)^{6} \right] + \sum_{\text{nonbond}} \frac{q_i q_j}{\epsilon_1 r_{ij}}
\end{aligned}
$$

where K_b, K_{ub}, K_θ, K_ϕ, and K_{ϕ_i} are force constants of bond, Urey-Bradley (UB) 1-3 distance, angle, dihedral angle, and improper dihedral angle potentials, respectively; b_0, S_0, θ_0, ϕ_0, and $\phi_{i,0}$ are corresponding equilibrium values; and δ is a phase shift of the dihedral angle potential. ϵ is a 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 determined according to the atom types involved.

The form of potential energy function is force field (FF) dependent. In **GENESIS**, TIP3P explicit water *[23]* with CHARMM22, CHARMM27, and CHARMM36 force fields are available (proteins: *[7][8]*, nucleic acids: *[24][25]*, lipids: *[26][27]*, and CHARMM36 *[28] [29]*).

In addition to CHARMM FF, AMBER FF *[6]* is also supported. As for AMBER FF, in addition to the original AMBER scheme, the scheme by Gromacs package is also implemented. The calculation styles for the dispersion correction term and the truncation of nonbonding term are quite different between AMBER and GROMACS packages. You can choose the style by setting *Forcefield* to *AMBER* or *GROAMBER*.

As for coarse-grained simulations, MARTINI (*[30] [31]*) and some structure-based models (GO-models *[32]*) are implemented. In the structure-based models, Off-latice GO models with Cα (*[33]*), all atom models (*[34]*), and a model proposed by Karanicolas and Brooks (*[35] [36]*) are supported.

---

forcefield *CHARMM / AMBER / GROAMBER / GROMARTINI / KBGO / CAGO / AAGO*

> Type of force field used for energy/force evaluation. GO-like models (*KBGO, CAGO, AAGO*) are available only in **ATDYN**. (**Default : CHARMM**)

- CHARMM: CHARMM all-atom force field
- AMBER: AMBER all-atom force field
- GROAMBER: AMBER all-atom force field (Gromacs scheme)
- GROMARTINI: MARTINI force field *[30] [31]*
- KBGO: model by Karanicolas and Brooks *[35] [36]*
- CAGO: Cα GO model *[33]*
- AAGO: all heavy atom GO model *[34]*

5.2 Non-bonded interaction related keywords

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

Simple spherical truncation at the cut-off value leads to discontinuous 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 for (1) potential energy, and (2) forces. The smooth potential energy switching function is enabled in the default setting (not available in AMBER FF, though) in **GENESIS**. To enable switch function that smoothly turns off forces *[37]*, "vdw_force_switch=YES" should be set.

Many of the force fields have been developed with their own MD packages. As a result, these force fields have 'their own' switching functions. GENESIS use different switching functions for different force fields. Therefore, the related options are not supported in some force fields. (For example, no switching functions are available for "forcefield=AMBER".) The use of defaults for each force field is highly recommended.

The spherical truncation of electrostatic energy is different from van der Waals due to its long-range nature. To truncate electrostatic energy at the cut-off value, a shift function is introduced which is enabled by "Electrostatic=Cutoff" in **GENESIS**. In case of the complete electrostatic energy, the smooth particle mesh Ewald (PME) method is used (it is explained in the PME-related keyword section).

---

electrostatic *CUTOFF/PME*

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

switchdist *Real*

Switch-on distance for nonbonding interaction energy/force quenching (unit : Angstrom). (**Default : 10.0**)

To disable switching, *switchdist* has to be set to the same value of *cutoffdist*.

Switching scheme depends on the selected force field, *vdw_shift*, and *vdw_force_switch* parameters. In case of AMBER force field, this switching must be disabled since switching is not available.

In case of `forcefield = GROMARTINI` and `electrostatic = CUTOFF`, *switchdist* is used only in van der Waals potential energy. The switching-on distance for the electrostatic energy is automatically defined as 0.0.

cutoffdist *Real*

Potential energy/force cut-off distance (unit : Angstrom). (**Default : 12.0**) This distance must NOT be smaller than *switchdist* above. In case of AMBER force field, this value must be equal to *switchdist*.

pairlistdist *Real*

Cut-off distance of Verlet pair list for non-bonded energy/force evaluations *[38]* (unit : Angstrom). (**Default : 13.5**) This distance must be larger than *cutoffdist*.

dielec_const *Real*

Dielectric constant of the system. (**Default : 1.0**)

vdw_force_switch *YES / NO*

This paramter determines whether the force switch function for van der Waals interactions is employed or not. (**Default : NO**) This parameter is available only when "forcefield = CHARMM". If you use CHARMM36 force field, you should turn on this switch.

vdw_shift *YES / NO*

This parameter determines whether the van der Waals potential energy shift is employed or not. If turned on, potential energy at the cut-off distance is shifted by a constant value so as to nullify the energy at that distance, instead of the default smooth quenching function. This parameter is available only when `forcefield = GROAMBER` or `forcefield = GROMARTINI`. (**Default : NO**)

dispersion_corr *NONE / ENERGY / EPRESS*

Determines the way of long-range corrections for the cut-off of van der Waals interactions. (**Default : NONE** (except for forcefield = AMBER, where only *EPRESS* is available.))

- **NONE**: no correction
- **ENERGY**: only the energy correction is considered.
- **EPRESS**: in addition to the energy correction, internal pressure (virial) correction is also considered.

The correction formula is different between GROMACS and AMBER styles. This parameter is not available for CHARMM force field, and *EPRESS* is always employed for AMBER force field.

contact_check *YES / NO*

(If you are using **SPDYN**, please check *structure_check* below.) If set to *YES*, bond distance and nonbonding contact checkers are activated. (**Default : NO**)

If this parameter is enabled, *nonb_limiter* below is also enabled. To enable this paramter only, please explicitly specify "contact_check = YES" and "nonb_limiter = NO".

The bond distance checker examines the distances of bonding pairs. On the other hand, nonbonding contact checker searches for close contacted non-bonded pairs. If suspiciously long-distanced bonds or too close nonbonding atom pairs are found, indexes of atoms involved are written in the log file. Those distance checks are performed during the setup.

structure_check *NONE / FIRST / DOMAIN* (**SPDYN only**)

If this option is enabled, covalent bond length check and nonbonding contact check will be performed on pairlist update. This function is very similar to *contact_check* above but has an improved capability especially if parallel-io scheme is employed, since *contact_check* does not work for parallel-io scheme. In **SPDYN**, we recommend users to use this option instead of *contact_check*. (**DEFAULT** : NONE)

- **NONE**: do not perform check (default)
- **FIRST**: perform check only at the beginning of the simulation
- **DOMAIN**: permform checks every time when updating pairlist

Please note that enabling this function will demand some additional computational costs. You should avoid the use of this option on your production runs.

nonb_limiter *YES / NO*

If this option is enabled, large forces due to short contacts are tempered during simulations by introducing a minimum contact distance (see *minimum_contact* below). (**Default : same as contact_check value**; *if "contact_check = YES" is specified in the control file, the default of this parameter also becomes "YES".*)

If you want to enable *contact_check* only, please specify explicitly both of "contact_check = YES" and "nonb_limiter = NO".

This tempering may be greatly useful in the system setup and equilibration stages.

Important Notice: we recommend users to use the option during minimization and first MD run to avoid memory lookup error due to the short contacts. However, you are not recommended to use this check in your production MD runs since this parameter might change the energies and forces in an unphysical way. If your MD run requires this option in the production stage, your system might have severe problems (such as ring-penetration). Please check your system carefully in those cases.

minimum_contact *Real*

(Active only if `contact_check = YES`) This parameter defines the minimum distance for nonbonding atom pairs. If a distance between a non-bonding atom pair is less than this value, energy/force will be evaluated using this distance rather than the actual distance. (unit : Angstrom) (**Default : 0.5**)

output_style *GENESIS / CHARMM / NAMD / GROMACS*

Energy output format. (**Default : GENESIS**)

5.3 Particle mesh Ewald related keywords

Electrostatic energy in the conventional Ewald sum method is expressed as:

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

Here, the cut-off scheme can be used for the first term, because it decreases rapidly as distance between atoms increases. The third term is so called *self-energy*, and is calculated only once. The second term can be 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 the structure factor $\mathbf{S}(\mathbf{G})$ is defined as:

$$\mathbf{S}(\mathbf{G}) = \sum_i q_i \exp(i\mathbf{G} \cdot \mathbf{r}_i)$$

We cannot employ fast Fourier transformation (FFT) for the calculation of $\mathbf{S}(\mathbf{G})$ since atomic positions are usually not equally spaced. In the smooth particle mesh Ewald (PME) method *[39] [40]*, this structure factor is approximated by using cardinal B-spline interpolation as:

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

where $b_1(G_1)$, $b_2(G_2)$, and $b_3(G_3)$ are the coefficients brought by the cardinal B-spline interpolation of order n and $\mathbf{Q}$ is a 3D tensor obtained by interpolating atomic charges on the grids. Since this $\mathbf{Q}$ has equally spaced structure, its Fourier transformation, $\mathbf{F}(\mathbf{Q})$, can be calculated by using FFT in the PME method.

The following keywords are relevant if `electrostatic=PME`.

pme_alpha *Real or auto*

> Exponent of complementary error function. If `pme_alpha=auto` is specified, the value is automatically determined from the *cutoffdist* and *pme_alpha_tol*. (**Default : auto**)
>
> Note: the default value of 0.34 was set in the former releases (version < 1.1.0). Please be careful when you compare results against former versions.

pme_alpha_tol *Real*

> (Active only if `pme_alpha=auto`) Tolerance used when determining *pme_alpha*. (**Default : 1.0e-5**)

pme_max_spacing *Real*

> Max PME grid size used in the automatic grid number determination (unit : Angstrom). (**Default : 1.2**)
>
> This parameter is used only when *pme_ngrid_x*, *pme_ngrid_y*, and *pme_grid_z* are not given in the control file.

pme_ngrid_x *Integer*

> Number of FFT grid points along x dimension. If not specified, program will determine an appropriate number of grids using `pme_max_spacing`. (**Optional**)

pme_ngrid_y *Integer*

> Number of FFT grid points along y dimension. If not specified, program will determine an appropriate number of grids using `pme_max_spacing`. (**Optional**)

pme_ngrid_z *Integer*

Number of FFT grid points along z dimension. If not specified, program will determine an appropriate number of grids using `pme_max_spacing`. (**Optional**)

pme_nspline *Integer*

B-spline interpolation order used for the evaluation of $b_1(G_1)$, $b_2(G_2)$, $b_3(G_3)$, and **Q**. The order must be $>= 3$. (**Default : 4**)

pme_multiple *YES/NO* (**ATDYN only**)

Enable partitioning of processes for PME real and reciprocal term computations (available only in **ATDYN**). (**Default : NO**)

pme_mul_ratio *Integer* (**ATDYN only**)

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

FFT_scheme *1DALLGATHER / 1DALLTOALL / 2DALLTOALL* (**SPDYN only**)

This is a highly advanced option concerning reciprocal space calculations. Users usually don't need to change this option. See ref *[41]* for details. (**Default : 1DALLTOALL**)

5.3.1 Number of PME grid points

Both of **ATDYN** and **SPDYN** use OpenMP/MPI hybrid parallel fast Fourier transformation library, FFTE *[42]*. The number of PME grid points must be multiples of 2, 3, and 5 due to the restriction of this library. Moreover, in **SPDYN**, there are several additional rules, which depends on the number of processes, in PME grid 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. The restriction condition of the grid numbers are as follows:

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`

If the given number of PME grid points does not meet the above conditions, the program will automatically reassign suitable grid numbers which are larger than those written in the control input. In such cases, warning message will be shown in the log file.

5.4 Lookup table related keywords

The following keywords are relevant if all atom force field is used. For a linearly-interpolating lookup table, 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

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

Linear interpolation is used if ''Electrostatic=PME''.

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 *[43]* using four basis functions for the cubic Hermite spline : $h_{00}(t)$, $h_{10}(t)$, $h_{01}(t)$, $h_{11}(t)$

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

where

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

and

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

Cubic iterpolation is used if ''Electrostatic=Cutoff''.

---

table_density *Integer*

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

water_model *expression or NONE*

Type of water model (i.e. name of water residues) used for the lookup table approach. In most cases, this parameter does not need to be changed. (**Default : NONE**)

But please be careful about another *water_model* parameter in the **[CONSTRAINTS]** section. This value should be modified according to the name of water molecule in the selected force field.

The default value has been changed in version 1.1.0.

CHAPTER

SIX

DYNAMICS

In the **[DYNAMICS]** section, we can choose integrator and various output update frequencies used in the MD simulations.

6.1 General keywords

In **ATDYN**, two integerators, leapfrog and velocity Verlet, are available. On the other hand, in **SPDYN**, multiple time step integrator named r-RESPA is available in addition to the leapfrog and velocity Verlet integrators. You should also check the **[ENSEMBLE]** section because integrators are intimately related to the thermostat/barostat specification. Actually, some of integrator-thermostat/barostat combinations are not available. See also *Ensemble* section for the details about this issue.

---

integrator *LEAP / VVER / VRES*

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

- **LEAP**: the leapfrog integrator
- **VVER**: the velocity Verlet integrator
- **VRES**: RESPA integrator (available only in **SPDYN**). *Not available in REMD and String method*.

nsteps *Integer*

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

timestep *Real*

MD time step (unit : ps). (**Default : 0.001** (1.0 fs))

eneout_period *Integer*

Frequency of energy outputs (unit : number of MD steps). (**Default : 10**)

If `timestep=0.001` (1 fs) and `eneout_period=10` (default), the energy log will be written every 10 fs.

crdout_period *Integer*

Frequency of coordinates (trajectory) outputs (unit : number of MD steps). (**Default : None**)

velout_period *Integer*

Frequency of velocity outputs (unit : number of MD steps). (**Default : None**)

rstout_period *Integer*

Frequency of restart file updates (unit : number of MD steps). (**Default : None**)

stoptr_period *Integer*

Frequency of system translation and rotation motion removals (unit : number of MD steps). (**Default : 10**)

Notice: removing translational motion might cause some unexpected behavior of the system. For example, positional (POSI) or rmsd (RMSD[MASS]) restraint could yield translational momentum of the system center-of-mass. If the translational momentum of the system center-of-mass is completely removed in those cases, the dynamics can be significantly disturbed (especially in coarse-grained systems). Users should carefully check the trajectory if you employ both of "stoptr" and POSI/RMSD[MASS] restraint. However, usually, positional restraints in the system equilibration stage may not suffer from this issue.

nbupdate_period *Integer*

Frequency of the non-bonded pair-list updates (unit : number of MD steps). (**Default : 10**)

elec_long_period *Integer* (**SPDYN only**)

(RESPA integrator only) Frequency of slow motion force evaluation (unit : number of MD steps). (**Default : 2**; every 2 steps)

thermostat_period *Integer* (**SPDYN only**)

(RESPA integrator only) Frequency of thermostat integration (unit : number of MD steps). (**Default : 1**)

It must be multiple of elec_long_period.

barostat_period *Integer* (**SPDYN only**)

(RESPA integrator only) Frequency of barostat integration (unit : number of MD steps). (**Default : 1**)

It must be multiple of thermostat_period.

iseed *Integer*

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

Note: if *iseed* is not specified in the restart run, the seed value in the restart file will be used.

initial_time *Real*

Initial time of this run (unit : ps). (**Default : 0.0**)

verbose *YES/NO*

Determines whether verbose output is enabled or not. (**Default : NO**)

6.2 Simulated annealing related keywords

The following keywords are relevant for MD simulations only. This function is not available in *VVER* and *VRES* integrators of **SPDYN**, please use *LEAP* integrator instead.

annealing *YES/NO*

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

anneal_period *Integer*

Frequency of temperature updates (unit : number of MD steps). (**Default : None**)

dtemperature *Real*

Temperature change in each temperature update (unit : Kelvin); the total temperature change in a MD run is `dtemperature * nsteps / anneal_period`. (**Default : 0.0**).

6.3 Targeted MD and Steered MD related keywords

In GENESIS, there are targeted MD (TMD) and steered MD (SMD) to guide a system toward target structure. In SMD, we assign restraint forces (steering forces) along the target RMSD value which is regularly changed during MD. At each time step, the restraint force on each atom is calculated from the derivative of the restraint RMSD potential energy:

$$U = \frac{1}{2}k\left(RMSD(t) - RMSD_0(t)\right)^2$$

where $RMSD(t)$ is the instantaneous RMSD of the current coordinates from the target coordinates and $RMSD_0$ is the target RMSD value at the given time t. Target RMSD value changes linearly with time from the initial RMSD to the target RMSD value:

$$RMSD_0(t) = RMSD_{\text{initial}} + \frac{t}{T}\left(RMSD_{\text{final}} - RMSD_{\text{initial}}\right)$$

where T is the total MD simulation time. Targeted MD (TMD), originally suggested by J. Schlitter et al. *[44]*, is different from SMD in that force constants are changed during MD simulations. If you perform SMD, there is a possibility observing the large difference between the instantaneous RMSD and target RMSD. In TMD, force constants are given by Lagrangian multipliers to overcome the energy barrier between the instantaneous and target RMSDs. Therefore, we could find trajectories where RMSD is almost identical to the target RMSD at each time. In **[SELECTION]** section, users can select atoms involved in RMSD calculations for both SMD and TMD. Users should specify either *RMSD* or *RMSDMASS* (mass-weighted RMSD) in **[RESTRAINTS]** section to run TMD or SMD. In SMD, force constants defined in **[RESTRAINTS]** section are used, but force constants are automatically determined using Lagrangian multipliers during simulation in TMD.

---

target_md *YES/NO*

Enabling targeted MD. (**Default : NO**)

steered_md *YES/NO*

Enabling steered MD. (**Default : NO**)

initial_rmsd *Real*

Initial value of reference rmsd (unit : Angstrom). If not specified explicitly, calculated from the initial and reference structures. (**Default : 0.0**)

Important Notice (1.1.5 or later) Structural fitting method should be defined in **[FITTING]** section (*Fitting*). The default behavior was significantly changed on 1.1.5 (no fitting applied on the default setting), so users of 1.1.4 or before must pay special attention on the fitting scheme. In versions of 1.1.4 or before, structural fitting is automatically applied for the atoms concerning restraint potential.

final_rmsd *Real*

Final value of reference rmsd (unit : Angstrom). (**Default : 0.0**)

CHAPTER

SEVEN

MINIMIZE

In **[MINIMIZE]** section, you can choose options for energy minimization. Currently, only the steepest descent (SD) algorithm is available.

---

method *SD*

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

nsteps *Integer*

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

eneout_period *Integer*

Frequency of energy outputs (unit : number of minimization steps). (**Default : 10**)

On default (`eneout_period=100`), the energy log will be written every 100 steps.

crdout_period *Integer*

Frequency of coordinates outputs (unit : number of minimization steps). (**Default : N/A**)

rstout_period *Integer*

Frequency of restart file updates (unit : number of minimization steps). (**Default : N/A**)

nbupdate_period *Integer*

Frequency of non-bonded pair-list updates (unit: number of minimization steps). (**Default : 10**)

force_scale_min *Real*

Minimum value of the force scaling coefficient in the steepest descent method. This value is also used as the initial value of the scaling coefficient. (**Default : 0.00005**)

force_scale_max *Real*

Maximum value of the force scaling coefficient in the steepest descent method. (**Default : 0.0001**)

Note: It is often the case that the initial structure has very short nonbondded contacts. Such unrealistic interactions often cause the crash of program. The use of option `contact_check=YES` (in **[ENERGY]** section) is recommended for the minimization run at early equilibration stage to avoid huge energy/force values and crashes due to memory lookup errors.

Note: Constraints such as SHAKE are NOT available in minimization. Restraints (such as positional restraint; *POSI*) are available.

CHAPTER

EIGHT

CONSTRAINTS

In the **[CONSTRAINTS]** section, keywords related to constraint of atoms can be specified. The SHAKE scheme is used for bonds between heavy and hydrogen atoms *[45]*. For velocity Verlet integrator, the RATTLE constraint scheme is used *[46]*. Bond constraints between heavy atoms are not available for now. The SETTLE algorithm *[47]* is available for water molecules, where the name of water molecule has to be given as *water_model* parameter.

---

rigid_bond *YES/NO*

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

fast_water *YES/NO*

Use the SETTLE algorithm for the constraints of the water molecules. (**Default : YES**)

shake_iteration *Integer*

Maximum number of iterations for SHAKE/RATTLE constraint. If SHAKE/RATTLE does not converge within the given number of iterations, the program terminates with an error message. (**Default : 500**)

shake_tolerance *Real*

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

water_model *expression or NONE*

Type of water model (i.e. name of water molecule) constrained by SETTLE. (**Default : TIP3**)

Important notice: When you are trying to use AMBER force field, please manually specify "water_model = WAT". Otherwise, programs might crash with an error.

hydrogen_type *NAME/MASS/BOTH*

This parameter defines how hydrogen atoms are detected. This parameter is ignored when *rigid_bond = NO*. Usually, users do not need to care about this parameter. (**Default : NAME**)

- **NAME**: determines from the atomic name, type, and mass. If the mass of an atom is larger than the that of a deuterium atom and the name or type begins with 'h', 'H', 'd', or 'D', that atom is considered as a hydrogen. In addition, if *NAME* is selected and suspicious light atoms are found, program will put an error and quit. A suspicious light atom has a mass smaller than 2.1 and the first characters of the name and type are NOT 'h', 'H', 'd', or ,'D' (for example, XX in the table below).

- **MASS**: determines only from the atomic mass.
- **BOTH**: determines from both the atomic mass and name. Please use it

for TIP4P water model case

Table. Relation of atomic name, mass, and flags.

atom name (type)	mass	*NAME*	*MASS*
HX	1.0	o	o
XX	1.0	x(*crash*)	o
HY	3.0	x	x
YY	3.0	x	x

o: considered as a hydrogen, x: not considered as a hydrogen.

hydrogen_mass_upper_bound *the upper bound to decide the hydrogen atom*

This parameter defines the upper bound to determine the hydrogen atom. For exmaple, if you define it as 3.0, then atoms with the atomic mass less than 3.0 are considered as hydrogen atoms. You should write it in the case of hydrogen mass repartitioning scheme. This is applied from **GENESIS 1.2**

fast_bond *YES/NO* (**ATDYN only**)

Determines whether LINCS is employed (**Default : NO**). To enable LINCS, you should also enable *rigid_bond* above.

lincs_iteration *Integer* (**ATDYN only**)

Iteration count for LINCS. (**Default : 1**)

lincs_order *Integer* (**ATDYN only**)

Matrix expansion order in LINCS. (**Default : 4**)

Note: TIP4P water model is applied from **GENESIS 1.2**. In the case of using TIP4P water model, we regard it as rigid. In molecular dynamics simulations, please define **rigid_bond** and **fast_water** yes. In minimization, **[Constraints]** has not been used before, but now you can define **fast_water** yes when TIP4P water model is used, by regarding TIP4P water molecule rigid. However, please keep in mind that other parameters cannot be defined in minimizations, and constraints are not applied except water molecules. TIP4P water model can be used only in SPDYN.

CHAPTER

NINE

ENSEMBLE

In the **[ENSEMBLE]** section, the type of ensemble, temperature and pressure control algorithm, and parameters used in these algorithms (such as temperature and pressure) can be specified.

In the Langevin thermostat algorithm ("ensemble=NVT" with "tpcontrol=LANGEVIN"), every particles are coupled with a viscous background and a stochastic heat bath *[48]*:

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

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 motion is given by *[49]*:

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

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 (XY), pressure along the normal (Z), temperature *[50]*, where *isotropy* (see below) must be 'XY-FIXED'.
- **NPgT**: Constant surface-tension (XY), pressure along the normal (Z), temperature *[50]*, where *isotropy* (see below) must be 'SEMI-ISO'.

temperature *Real*

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

pressure *Real*

Target pressure in NPT or target pressure along the 'Z' axis in NPAT and NPgT (unit : atm). (**Default : 1.0**)

gamma *Real*

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

tpcontrol *NO / BERENDSEN / LANGEVIN / BUSSI*

Type of thermostat and barostat (**Default : NO**).

- **NO**: Do not use temperature/pressure control algorithm (for NVE only)
- **BERENDSEN**: Berendsen thermostat/barostat *[51]*
- **LANGEVIN**: Langevin thermostat/barostat *[49]*
- **BUSSI**: Bussi's thermostat/barostat *[52] [53]*

Table. Available combinations of integrators and thermostats/barostats.

	LEAP			VVER			VRES (**SPDYN only**)*		
	NVT	NPT	NPAT/NPgT	NVT	NPT	NPAT/NPgT	NVT	NPT	NPAT/NPgT
BEREND-SEN	o	o	–	o	–	–	–	–	–
LANGEVIN	o	o	o	o	o	o	o	o	o
BUSSI	–	–	–	o	o	–	o	o	–

* VRES is not available for *REMD* and *String method*

---

tau_t *Real*

Temperature coupling time in Berendsen and Bussi thermostats (unit : ps). (**Default : 5.0**)

tau_p *Real*

Pressure coupling time in Berendsen and Bussi barostats (unit : ps). (**Default : 5.0**)

compressibility *Real*

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

gamma_t *Real*

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

gamma_p *Real*

Friction parameter of Langevin 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, and 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 can change independently *[54]*. This setting with NPT or NPAT or NPgT ensemble is expected to be useful for bio-membrane systems.
- **XY-FIXED**: X and Y dimensions are fixed, while Z dimension can change (NPAT only).

CHAPTER

TEN

BOUNDARY

In the **[BOUNDARY]** section, boundary conditions of the system such as simulation box size can be 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 only in ATDYN**).

box_size_x *Real*

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

This size will be replaced by the value in restart file (rstfile). Please see the note below.

box_size_y *Real*

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

This size will be replaced by the value in restart file (rstfile). Please see the note below.

box_size_z *Real*

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

This size will be replaced by the value in restart file (rstfile). Please see the note below.

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. When users specify the number of domains explicitly, please make sure that the product of domain numbers (domain_x * domain_y * domain_z) is 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 MD, the initial box size is read from the restart file and the values specified in the control file is ignored.

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, those atoms can be easily refered from **[RESTRAINTS]** section. Nothing is selected as default.

[SELECTION] section is also used in the control file of some analysis tools, in which the section is used to specify the atoms being analyzed and/or fitted.

In the following parameters, the last character 'N' has to be replaced by a positive integer number (i.e. $N \geq 1$), which defines the ID of the selection. This ID is used when referring from other sections such as **[RESTRAINTS]**. See also examples in the bottom of the section.

group*N* *expression*

> ('N' must be replaced by a positive integer value.) Select atoms by *expression* and define them as a group. Available keywords and operators in *expression* are listed in the table below. Note that mname (or moleculename, molname) in *expression* is a molecule name that is defined by `mol_name` below.

mole_name*N* *molecule starting-residue ending-residue*

> ('N' must be replaced by a positive integer value.) Define *molecule* by *starting-residue* and *ending-residue*. Those residues are defined by
>
> `[segment id]:residue number:residue name`

Table. 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[*1]	ai:1-5	atomindex, atomidx
atno:*number[-[number]]*	atom number[*1]	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
hydrogen	hydrogen atoms		hydrogenatom
heavy	heavy atoms		heavyatom
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 wants to use such PDB file as an input file, although it should 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) | an:CA
mole_name1 = molA PROA:1:TYR PROA:5:MET
group3     = mname:molA and (an:CA or an:C or an:O or an:N)
```

CHAPTER

TWELVE

RESTRAINTS

[RESTRAINTS] section contains keywords to define external restraint functions. The restraint functions are applied to the selected atom groups in **[SELECTION]** section to restrict the motions of those atoms.

The potential energy of a restaint can be written as:

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

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

Note: (**SPDYN only**) The restraints in this section are different from the local restraint functions. Please also check *localresfile* in **[INPUT]**. (*Input and Output files*).

Keywords in this section, except **nfunctions**, **pressure_position**, and **pressure_rmsd**, must have a serial number, 'N', of the function ($N \geq 1$). This serial number is referred when selecting restraints in REUS runs. See example inputs in the end of this section. Keywords of **pressure_position** and **pressure_rmsd** are used when the external virial due to position and rmsd restraints are included in instantaneous pressure evaluations.

pressure_position *YES/NO*

The virial terms from position restraints are included in pressure evalutions. (**Default : NO**)

pressure_rmsd *YES/NO*

The virial terms from rmsd restraints are included in pressure evalutions. (**Default : NO**)

nfunctions *Integer*

Number of restraint functions. Note that number of local restraints must NOT be included here. (**Default: 0**)

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

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

- *POSI*: positional restraint. The reference coordinates are given by **reffile**, **ambreffile**, or **groreffile** in **[INPUT]**. (see *Input and Output files*)
- *DIST[MASS]*: distance restraint.

- *ANGLE[MASS]*: angle restraint.
- *DIHED[MASS]*: dihedral angle restraint.
- *RMSD[MASS]*: RMSD restraint. *MASS* means mass-weighted RMSD. Translational and rotational fitting to the reference coordiate are done before calculating RMSD. The reference coodinate is specified in the same manner as *POSI*.

  Important Notice (1.1.5 or later) Structural fitting method can be defined in **[FITTING]** section (*Fitting*) on 1.1.5 or later. Users of GENESIS 1.1.4 or before should pay special attention on the fitting scheme. In versions 1.1.4 or before, translational and rotational fittings were automatically applied for the atoms concerning RMSD restraint. (same as the current default setting, *fitting_method* = *TR+ROT*)
- *PC[MASS]*: principal component constraint. *modefile* in the *Input and Output files* section is required to use this constraint.

DIST, *ANGLE*, *DIHED* impose restraint on distance/angle/dihedral defined by the selected groups. See `select_indexN` and examples below for the specification. *MASS* indicates that the force is applied to the center of mass of the selected group. When *MASS* is omitted, the force is applied to the geometric center of the coordinates. *MASS* keyword does nothing for groups consist of a single particle.

In **SPDYN**, *POSI* and *RMSD[MASS]* restraints are mutually exclusive; you can use either one or none of them. Two different *POSI* restraints might not be applied simultaneously, either.

Notice: POSI, PC, and RMSD restraints can be influenced by the removal of translational/rotational momentum. See also the notices in the stoptr_period parameter in the [DYNAMICS] section.

constant*N* *Real*

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

The unit of *DIST/RMSD* is kcal/mol/Angstrom^n and that of *ANGLE/DIHED* is kcal/mol/rad^n, where n is *exponentN* in this section.

If you employed a certain restraint term for REUS runs, *nreplica* of force constants must be given as a space-separated list. See examples in the *REMD* section.

reference*N* *Real*

Reference value of a restraint function. For the positional 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^n, where *n* is *exponentN*.

If you employed a certain restraint term for REUS runs, *nreplica* of reference values must be given as a space-separated list. See example in the *REMD* section.

select_index*N* *Integer*

Index of an atom group, to which restraint potentials are applied. The index must be defined in **[SELECTION]** (see *Selection*). For example, if you specify `select_index1 = 1`, this restraint function is applied for `group1` in the **[SELECTION]**. (**Default: N/A**)

Number of groups required depends on the type of the restraint function.

- *POSI/RMSD[MASS]*: 1

- *DIST[MASS]*: $2m$, where $m = 1, 2, ...$
- *ANGLE[MASS]*: 3
- *DIHED[MASS]*: 4
- *PC[MASS]*: $>= 1$

A group can contain more than single atom. Suppose we have the following input.

```
[SELECTION]
group1        = ai:1-10
group2        = ai:11-20

[RESTRAINTS]
nfunctions    = 1
function1     = DIST
constant1     = 3.0
reference1    = 10.0
select_index1 = 1 2
```

In this case, the distance restraint is applied for the distance between geometric centers of group1 and group2. The calculated force is then scattered to each atom. If *DISTMASS* is given instead of *DIST*, mass centers (mass-weighted average position) are used instead of geometric centers (not mass-weighted average position).

In case of *DIST[MASS]* restraint with more than 2 groups specified (i.e. $2m$ with $m \geq 2$), the sum of m distances will be restrained. See `exponent_dist` and `weight_dist` parameters for this distance summation. However, this scheme might not be useful for the standard cases.

direction*N* *ALL / X / Y / Z*

Direction of the *POSI* restraint. If X/Y/Z is specified, restraints along the other two axes are not applied. (**Default : ALL**)

exponent*N* *Integer*

Exponent factor of the restraint function. The default is the harmonic. (**Default : 2**)

This parameter does not work for *POSI* and *RMSD[MASS]* restraints in **SPDYN**, where the default value, 2, is always used.

exponent_dist*N* *Integer*

(*DIST[MASS]* only) Exponent factor used in the distance sum calculations. (**Default : 1**)

The sum of distances is expressed as: $r_{\mathrm{sum}} = \sum_m w|r_m|^n$, where $(1 \leq m \leq \mathrm{num\ groups}/2)$, n is *exponent_distN*, and w is *weight_distN*.

weight_dist*N* *Real*

(*DIST[MASS] only*) Weight factor used in the distance sum calculations. (**Default : 1.0**)

mode*N* *Integer*

Specifies the mode index which is used for the PC (principal component) restraint. For example, the 1st PC mode can be restrained by specifying `mode1=1`.

Example of **[RESTRAINTS]** section:

```
[RESTRAINTS]
nfunctions    = 1
function1     = DIST
reference1    = 10.0
constant1     = 2.0
select_index1 = 1 2         # group1 and group2 in [SELECTION]
```

Example of multiple restraints:

```
[RESTRAINTS]
nfunctions    = 2

function1     = DIST
constant1     = 2.0
reference1    = 10.0        # in angstrom
select_index1 = 1 2

function2     = DIHED
constant2     = 3.0
reference2    = 120.0       # in degrees
select_index2 = 3 4 5 6
```

CHAPTER

THIRTEEN

FITTING

(GENESIS 1.1.5 or later only) Keywords in **[FITTING]** section define structure superimposition scheme, which is often employed in targeted MD, steered MD, or *String method* with positional restraint. In *String method*, the reference coordinate for fitting is given by **fitfile** in **[INPUT]** section. Otherwise (MD, MIN, REMD), the reference coordinate is given by **reffile**, **ambreffile**, or **groreffile** in **[INPUT]** section.

---

fitting_method *NO/TR+ROT/XYTR+ZROT*

Type of fitting method.

- NO: No fitting routine is applied
- TR+ROT: Remove both of translation and rotation
- XYTR+ZROT: Remove translation in XY-plane and rotation along the Z-axis

(**Default: TR+ROT**)

fitting_atom: *Integer*

Index of an atom group which is to be fitted to the reference structure. In RMSD restraints, Steered MD, or Targeted MD, this should be identical to the group where the restraint potential is applied.

The index must be defined in **[SELECTION]** (see *Selection*). For example, if you specify `fitting_atom = 1`, the reference atoms are members of `group1` in the **[SELECTION]**. (**Default: N/A**)

mass_weight: *YES / NO*

If the parameter is set to *YES*, mass-weighted fitting is employed. This parameter should be *YES* for RMSDCOM/PCCOM restraints and should be *NO* for RMSD/PC restraints. Please make sure that this parameter is correctly specified when you perform RMSD/RMSDCOM/PC/PCOM type of calculations. In *String method*, mass-weighted superimposition is not supported. (**Default: NO**)

force_no_fitting: *YES / NO*

This parameter must not be changed for standard MD runs. Translational and rotational fittings are usually required to calculate correct RMSD values. So GENESIS simulators (ATDYN and SPDYN) do not allow *fitting_method = NO* for calculations involving RMSD evaluation (targeted/steered MD, for example). But such fitting is not desirable when generating initial structure set for *String method* calculations using Cartesian coordinate as CV. Actually, *fitting_method = NO* was implemented just for this specific purpose. If you are

really want to turn off fittings for preparation of initial structure set for String method using RMSD restraint, please specify *fitting_method = NO* and *force_no_fitting = YES*.

Please make sure that *force_no_fitting = NO* is not always required when preparing initial structure set for *String method*. (**Default: NO**)

Example of **[FITTING]** section

```
[FITTING]
fitting_method = TR+ROT
fitting_atom   =  1
mass_weight    =  NO
```

CHAPTER

FOURTEEN

REMD

In the **[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 one of the most widely used methods in biomolecules' simulations *[20] [55]*. Here, replicas (or copies) of the original system are prepared, and different temperatures are assigned to each replica. Each replica runs in a canonical (NVT) or isobaric-isothermal (NPT) ensemble, and the temperatures are periodically exchanged between the neighboring replicas during a simulation. Exchanging temperature enforces a random walk in temperature space, allowing the system overcoming energy barriers and sampling much wider conformational space.

In REMD methods, the transition probability of 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 the algorithms used: i.e. constant temperature and constant pressure algorithms. On the other hand, 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 *[56] [57]*. In GENESIS, barostat momentum is rescaled in the case of T-REMD with Langevin or Bussi method in NPT, NPAT, and NPgT ensembles. For the other cases, only atomic momenta are rescaled.

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

REMD simulations in GENESIS require an MPI environment. At least one MPI process must be assigned to one replica. For example, when user wants to simulate 32 replicas, $32n$ MPI processes are required.

In the following parameters excluding *dimension*, *exchange_period*, and *iseed*, the last character 'N' has to be replaced by a positive integer number (i.e. $N \geq 1$), which defines the index of replica dimension. For example, *type1*, *nreplica1* are the replica type and number of replicas for the first dimension, respectively. See also examples in the bottom of this section.

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 *Integer*

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

type*N* *TEMPERATURE / PRESSURE / GAMMA / RESTRAINT / REST*

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

- **TEMPERATURE**: Temperature REMD *[20]*
- **PRESSURE**: Pressure REMD *[58]*
- **GAMMA**: Surface-tension REMD *[59]*
- **RESTRAINT**: REUS (or Hamiltonian REMD) *[21] [60]*
- **REST**: replica exchange with solute tempering (REST2 or gREST) *[61] [62]*. (Note: this is different from the original version of REST *[63]*.) Currently, only AMBER and CHARMM force fields are supported.

nreplica*N* *Integer*

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

parameters*N* *Real*

List of parameters for each replica in the specified dimension. Parameters must be given as a space-separated list and the total number of parameters must be equal to the above *nreplicaN*. In case of REUS (`type = RESTRAINT`), *nreplicaN* of parameters must be given in **[RESTRAINTS]** section (not here).

See examples in the end of this section for details.

In case of *REST*, these parameters are considered as temperature of solute region.

Note that REUS input style has been changed on verion 1.1.0. The restraint parameters can no longer be given in this section.

rest_function*N* (available for *REUS* only)

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

Restrictions:

- *POSI* restraint is not available for REUS.
- *PC[MASS]* restraint is not available for REUS in **SPDYN**.

In addition to the normal REUS calculations, **GENESIS** supports so-called off-grid REUS calcutions. If you specify multiple restraints, those restraints will be merged into single reaction coordinate. Those restraints has to be defined in **[RESTRAINTS]** section, where the number of parameters (const and reference) must be equal to *nreplicaN*. Please check the example below. Note that this kinds of combined axis can be used only for restraints, other types (such as combined tempreature-pressure or temperature-restraint coordinate) are not available at least for now.

Note that REUS input style has been changed significantly on verion 1.1.0.

cyclic_params*N* *YES / NO*

If `cyclic_paramsN = YES` is specified, the first and last parameters in *parametersN* or *referenceN* in **[RESTRAINTS]** are considered as neighbouring parameters. (**Default: NO**)

Cyclic parameters are useful when REUS in dihedral angle space is carried out.

select_index*N* *Integer* (available for *REST* only)

Index of an atom group. The selected atoms are considered as "solute" in REST. The index must be defined in **[SELECTION]** (see *Selection*). (**Default: N/A**)

See also *param_indexN* for the detail about solute selection.

In SPDYN, water atoms cannot be specified as "solute" now. This limitation will be removed in the future version.

param_type*N* *ALL / BOND / ANGLE / UREY / DIHEDRAL / IMPROPER / CMAP / CHARGE / LJ* (available for *REST* only)

Solute energy terms for gREST *[62]* simulations. Energy terms selected by this parameter in the solute atom group (defined by *select_indexN*) are considered as "solute" (scaled according to solute temperature) in gREST. Other terms are considered as "solvent" (kept intact). Solute-solvent terms are automatically determined from the solute selection. You can specify multiple terms (see example below). (**Default: ALL**)

Note: parameter names below are case-insensitive.

- *ALL*: all the available energy terms (default).
- *BOND* (aliases: *B*, *BONDS*): 1-2 bonding terms.
- *ANGLE* (aliases: *A*, *ANGLES*): 1-2-3 angle terms.
- *UREY* (aliases: *U*, *UREYS*): Urey-Bradley terms.
- *DIHEDRAL* (aliases: *D*, *DIHEDRALS*): 1-2-3-4 dihedral terms.
- *IMPROPER* (aliases: *I*, *IMPROPERS*): improper torsion terms.
- *CMAP* (aliases: *CM*, *CMAPS*): CMAP terms.
- *CHARGE* (aliases: *C*, *CHARGES*): coulombic interaction terms (inc. PME reciprocal).
- *LJ* (aliases: *L*, *LJS*): Lennard-Jones interaction terms.

Note that restraint energy terms defined in **[RESTRAINTS]** cannot be solute terms. They never be affected by REST solute temperatures.

Note: When one dimensional T-REMD, P-REMD, and surface-tension REMD is carried out, parameters respectively specified by *temperature*, *pressure*, and *gamma* in **[ENSEMBLE]** section are ignored. For example, in the case of T-REMD in the NPT ensemble, 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 (type1 = TEMPERATURE and type2 = PRESSURE), temperature is exchanged first, followed by pressure exchange. This is repeated during the simulations.

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

Example of conventional T-REMD using 4 replicas:

```
[REMD]
dimension        = 1
exchange_period  = 1000
type1            = TEMPERATURE
nreplica1        = 4
parameters1      = 298.15 311.79 321.18 330.82
```

Example of two-dimensional REMD (temperature and Hamiltonian) using 32 ($8 * 4$) replicas:

```
[REMD]
dimension        = 2
exchange_period  = 1000

type1            = TEMPERATURE
nreplica1        = 8
parameters1      = 298.15 311.79 321.18 330.82 340.70 350.83 361.23 371.89

type2            = RESTRAINT
nreplica2        = 4
rest_function2   = 1

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

[RESTRAINTS]
nfunctions       = 1
function1        = DIST
constant1        =  2.0  2.0  2.0  2.0 # num of values must be nreplica
reference1       = 10.0 10.5 11.0 11.5
select_index1    = 1 2
```

Example of off-grid REUS (merge two restraints into single reaction coordinate), where distance and dihedral restraints are merged into single reaction coordinate. First values of restraints ((2.0,10.0) for distance, (10,-40) for dihedral) will be used for the first replica, the fourth parameters ((2.0,11.5) for distance, (10,-10) for dihedral) will be used for the fourth replica:

```
[REMD]
dimension        = 1
exchange_period  = 1000

type1            = RESTRAINT
nreplica1        = 4
rest_function1   = 1 2  # off-grid REUS

[SELECTION]
group1           = ai:1
group2           = ai:2
group3           = ai:3
group4           = ai:4
group5           = ai:5
group6           = ai:6

[RESTRAINTS]
nfunctions       = 2

function1        = DIST
constant1        =  2.0  2.0  2.0  2.0 # num of values must be nreplica1
reference1       = 10.0 10.5 11.0 11.5
select_index1    = 1 2

function2        = DIHED
constant2        =  10   10   10   10 # num of values must be nreplica1
reference2       = -40  -30  -20  -10
select_index2    = 3 4 5 6
```

Example of REST, where dihedral, CMAP, and LJ terms of certain atoms are selected as “solute”.

```
[REMD]
dimension        = 1
exchange_period  = 1000

type1            = REST
nreplica1        = 4
parameters1      = 300.0 310.0 320.0 330.0 # solute temperatures
param_type1      = D CM L  # dihedral, CMAP, and LJ
select_index1    = 1       # above terms inside this atom group are
                           # considered as "solute" terms

[SELECTION]
group1           = ai:1-313
```

CHAPTER

FIFTEEN

STRING METHOD

In the **[RPATH]** section, users can specify keywords for the string method. The string method is a powerful sampling technique which finds the most probable pathway (minimum free energy path; MFEP) connecting two stable conformational states. This method is often used for investigating large-scale conformational changes of biomolecules where time-scale of the transitions are not reachable in brute-force simulations.

There are three major algorithms in the string method: the mean forces string method *[64]*, the on-the-fly string method *[65]*, and the string method of swarms of trajectories *[66]*. Among these algorithgms, the mean forces string method is available in **ATDYN** and **SPDYN** *[67]*.

In the mean-forces string method, the pathway is represented by discretized points (called images) in the collective variable (CV) space. The current GENESIS supports distances, angles, dihedrals, Cartesian coordinates, and principal components for CVs (note that different types of CVs cannot be mixed in GENESIS. For example, users cannot mix distance and angle). In the calculation, each image is assigned to each replica, and a replica samples mean forces and an average metric tensor around its own image by short MD simulation (ps to ns length) with restraints. The restraints are imposed using the image coordinates as their reference values. After the short simulation , each image is evolved according to the mean force and metric tensor. Then, smoothing and re-parametrization of images are performed and go to the next cycle.

Image coordinates are written in rpath files (**rpathfile** keyword) which user can specify in **[OUTPUT]** section. This file provides the trajectory of image coordinates. Columns correspond to CVs and rows are time steps. These values are written at the same timing with **dcdfile** (specified by **crdout_period** in **[DYNAMICS]** section).

For the string method calculation, an initial pathway in the CV space and atomistic coordinates around the pathway are required. For preparing these, targeted or steered MD methods are recommended.

---

nreplica *Integer*

Number of replicas (images) for representing the pathway (**Default: 1**).

rpath_period *Integer*

Time-step period during which the mean-forces acting on the images are evaluated. After evaluating the mean-forces, the images are updated according to the mean-forces, then go to the next cycle. If `rpath_period = 0`, images are not updated. This option is used for equilibration or umbrella sampling around the pathway (**Default: 0**).

delta *Real*

Step-size for steepest descent update of images (**Default: 0.0**).

smooth *Real*

Smoothing parameter which controls the aggressiveness of the smoothing (**Default: 0.0**, no-smoothing). Values from 0.0 to 0.1 are recommended.

rest_function *List of Integers*

List of restraint function indices defined in **[RESTRAINTS]** section (see *Restraints*). Specified restraints are defined as CVs, and *nreplica* images (replicas) are created, where a set of corresponding restraint reference values is assigned to each image. Force constants in **[RESTRAINTS]** are also used for evaluation of mean-forces.

fix_terminal *YES / NO*

If `fix_terminal = YES` is specified, the two terminal images are always fixed and not updated. This is useful if the terminal images correspond to crystal structures and users do not want to move them (**Default: NO**).

use_restart *YES / NO*

Restart file generated by the string method calculation includes the last snapshot of images. If `use_restart = YES` is specified, the reference values in **[RESTRAINTS]** will be overwritten by the values in the restart file. Note that force constants are not overwritten. (**Default : YES**)

Obsolete parameters

The following options are move to **[FITTING]** section from GENESIS 1.1.5.

fitting_method *TR+ROT / XYTR+ZROT / NO*

This keyword is used only when CVs are Cartesian coordinates. If this keyword is specified, roto-translational elements are removed from the mean-force estimation by fitting instaneous structures to the reference coordinates given by `fitfile`.

fitting_atom *List of Integers*

This keyword is used only when CVs are Cartesian coordinates. The user can specify index of an atom group which are fitted to the reference structure. Usually, the same atoms as CVs are selected.

Example of alanine-tripeptide with 16 replicas (images). Two dihedral angles are specified as the collective variables.

```
[RPATH]
nreplica          = 16
rpath_period      = 1000
delta             = 0.02
smooth            = 0.0
rest_function     = 1 2

[SELECTION]
group1         = atomindex:15
group2         = atomindex:17
group3         = atomindex:19
group4         = atomindex:25
group5         = atomindex:27

[RESTRAINTS]
```

```
nfunctions    = 2

function1     = DIHED
constant1     = 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 \
                100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0
reference1    = -40.0 -40.0 -40.0 -40.0 -40.0 -40.0 -40.0 -40.0 \
                -40.0 -40.0 -40.0 -40.0 -40.0 -40.0 -40.0 -40.0
select_index1 = 1 2 3 4  # PHI

function2     = DIHED
constant2     = 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 \
                100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0
reference2    = -45.0 -33.0 -21.0 -9.0 3.0 15.0 27.0 39.0 \
                 51.0  63.0  75.0 87.0 99.0 111.0 123.0 135.0
select_index2 = 2 3 4 5  # PSI
```

Here is another example of Cartesian coordiante CVs for the same alanine-tripeptide.

```
[INPUT]
... skip ...
rstfile = ../eq/{}.rst
reffile = {}.pdb
fitfile = fit.pdb

[RPATH]
nreplica          = 16
rpath_period      = 1000
delta             = 0.001
smooth            = 0.00
rest_function     = 1
fix_terminal      = NO

[FITTING]
fitting_method = TR+ROT
fitting_atom   = 1

[SELECTION]
group1        = ai:15 or ai:17 or ai:19 or ai:25 or ai:27

[RESTRAINTS]
nfunctions    = 1

function1     = POSI
constant1     = 10.0 10.0 10.0 10.0 \
                10.0 10.0 10.0 10.0 \
                10.0 10.0 10.0 10.0 \
                10.0 10.0 10.0 10.0
select_index1 = 1
```

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.*, 30:1545–1614, 2009. URL: http://dx.doi.org/10.1002/jcc.21287, doi:10.1002/jcc.21287 (https://doi.org/10.1002/jcc.21287).

[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*, 29:845–854, 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.*, 26:1781–1802, 2005. URL: http://dx.doi.org/10.1002/jcc.20289, doi:10.1002/jcc.20289 (https://doi.org/10.1002/jcc.20289).

[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.*, 26:1668–1688, 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*, 102:3586–3616, 1998.

[8] A. D. MacKerell, M. Feig, and C. L. Brooks. Improved treatment of the protein backbone in empirical force fields. *J. Am. Chem. Soc.*, 126:698–699, 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.*, 118:11225–11236, 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.*, 25:1656–1676, 2004.

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

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

[19] SMOG: Structure-based Models for Biomolecules. http://smog-server.org/.

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

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

[22] Gromacs. http://www.gromacs.org/.

[23] 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.*, 79:926–935, 1983.

[24] 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.*, 21:86–104, 2000.

[25] 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.*, 21:105–120, 2000.

[26] 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.*, 73:2269–2279, 1997.

[27] 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*, 114:7830–7843, 2010.

[28] 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.*, 8:3257–3273, 2012.

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

[30] S.J. Marrink, H.J. Risselada, S. Yefimov, D.P. Tieleman, and A.H. de Vries. The MARTINI force-field: coarse grained model for biomolecular simulations. *J. Phys. Chem. B*, 111:7812–7824, 2007.

[31] L. Monticelli, S.K. Kandasamy, X. Periole, R.G. Larson, D.P. Tieleman, and S.J. Marrink. The MARTINI coarse grained forcefield: extension to proteins. *J. Chem. Theo. Comput.*, 4:819–834, 2008.

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

[33] C. Clementi, H. Nymeyer, and J. Onuchic. Topological and energetic factors: what determines the structural details of the transition state ensemble and "en-route" intermediates for protein folding? an investigation for small globular proteins. *J. Mol. Biol.*, 298:937–953, 2000.

[34] P. C. Whitford, J. K. Noel, S. Gosavi, A. Schug, K. Y. Sanbonmatsu, and J. N. Onuchic. An all-atom structure-based potential for proteins: Bridging minimal models with all-atom empirical forcefields. *Proteins: Structure, Function, and Bioinformatics*, 75:430–441, 2009.

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

[36] 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.*, 334:309–325, 2003.

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

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

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

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

[41] J. Jung, C. Kobayashi, T. Imamura, and Y. Sugita. Parallel implementation of 3D FFT with volumetric decomposition schemes for efficient molecular dynamics simulations. *Comp. Phys. Comm.*, 200:57–65, 2016.

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

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

[44] J. Schlitter, M. Engels, and P. Kruger. Targeted molecular dynamics: a new approach for searching pathways of conformational transitions. *J. Mol. Graph.*, 12:84–89, 1994.

[45] 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.*, 23:327–341, 1977.

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

[47] S. Miyamoto and P. A. Kollman. Settle - an Analytical Version of the Shake and Rattle Algorithm for Rigid Water Models. *J. Comput. Chem.*, 13:952–962, 1992.

[48] 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.*, 64:2375–2388, 1976.

[49] D. Quigley and M. I. J. Probert. Langevin dynamics in constant pressure extended systems. *J. Chem. Phys.*, 120:11432–11441, 2004.

[50] Y. Zhang, S. E. Feller, B. R. Brooks, and Pastor R. W. Computer simulation of liquid/liquid interfaces. I. Theory and application to octane/water. *J. Chem. Phys.*, 103:10252–10266, 1995.

[51] H. J. C. Berendsen, J. P. M. Postma, W. F. Vangunsteren, A. Dinola, and J. R. Haak. Molecular-Dynamics with Coupling to an External Bath. *J. Chem. Phys.*, 81:3684–3690, 1984.

[52] G. Bussi, D. Donadio, and M. Parrinello. Canonical sampling through velocity rescaling. *J. Chem. Phys.*, 126:014101, 2007.

[53] G. Bussi, T. Zykova-Timan, and M. Parrinello. Isothermal-isobaric molecular dynamics using stochastic velocity rescaling. *J. Chem. Phys.*, 130:074101, 2009.

[54] C. Kandt, W. L. Ash, and D. P. Tieleman. Setting up and running molecular dynamics simulations of membrane proteins. *Method*, 41:475–488, 2007.

[55] A. Mitsutake, Y. Sugita, and Y. Okamoto. Generalized-ensemble algorithms for molecular simulations of biopolymers. *Biopolymers*, 60:96–123, 2001.

[56] Y. Mori and Y. Okamoto. Generalized-ensemble algorithms for the isobaric-isothermal ensemble. *J. Phys. Soc. Jpn.*, 79:074003, 2010.

[57] Y. Mori and Y. Okamoto. Replica-exchange molecular dynamics simulations for various constant temperature algorithms. *J. Phys. Soc. Jpn.*, 79:074001, 2010.

[58] T. Okabe, M. Kawata, Y. Okamoto, and M. Mikami. Replica-exchange Monte Carlo method for the isobaric-isothermal ensemble. *Chem. Phys. Lett.*, 335:435–439, 2001.

[59] T. Mori, J. Jung, and Y. Sugita. Surface-tension replica-exchange molecular dynamics method for enhanced sampling of biological membrane systems. *J. Chem. Theory. Comput.*, 9:5629–5640, 2013.

[60] H. Fukunishi, O. Watanabe, and S. Takada. On the Hamiltonian replica exchange method for efficient sampling of biomolecular systems: Application to protein structure prediction. *J. Chem. Phys.*, 116:9058–9067, 2002.

[61] T. Terakawa, T. Kameda, and S. Takada. On Easy Implementation of a Variant of the Replica Exchange with Solute Tempering in GROMACS. *J. Comput. Chem.*, 32:1228–1234, 2011.

[62] M. Kamiya and Y. Sugita. Flexible selection of the solute region in replica exchange with solute tempering: Application to protein-folding simulations. *J. Chem. Phys.*, 149:072304, 2018.

[63] P. Liu, B. Kim, R. A. Friesner, and B. J. Berne. Replica exchange with solute tempering: A method for sampling biological systems in explicit water. *Proc. Natl. Acad. Sci. USA*, 102:13749–13754, 2005.

[64] L Maragliano, A. Fischer, E. Vanden-Eijnden, and G. Ciccotti. String method in collective variables: minimum free energy paths and isocommittor surfaces. *J. Chem. Phys.*, 125:24106, 2006.

[65] L. Maragliano and E. Vanden-Eijnden. On-the-fly string method for minimum free energy paths calculation. *Chem. Phys. Lett.*, 446:182–190, 2007.

[66] A. C Pan, D. Sezer, and B. Roux. Finding transition pathways using the string method with swarms of trajectories. *J. Phys. Chem. B*, 112:3432–3440, 2008.

[67] Y. Matsunaga, Y. Komuro, C. Kobayashi, J. Jung, T. Mori, and Y. Sugita. Dimensionality of Collective Variables for Describing Conformational Changes of a Multi-Domain Protein. *J. Phys. Chem. Lett.*, 7:1446–1451, 2016.