GENESIS User Guide

Release 2.0.0beta

RIKEN

Jun 17, 2020

GENESIS 2.0.0beta

Project Leader: Yuji Sugita (RIKEN)

Current main developers: Jaewoon Jung (RIKEN), Chigusa Kobayashi (RIKEN), Takaharu Mori (RIKEN), Kiyoshi Yagi (RIKEN), Hiraku Oshima (RIKEN), Yasuhiro Matsunaga (RIKEN/Saitama University)

Current developers/contributors: Osamu Miyashita (RIKEN), Florence Tama (RIKEN/Nagoya University), Motoshi Kamiya (RIKEN), Daisuke Matsuoka (RIKEN), Donatas Surblys (RIKEN), Kenta Yamada (RIKEN), Isseki Yu (RIKEN/Maebashi Institute of Technology), Suyong Re (RIKEN)

Other developers/contributors for older versions: Takao Yoda (Nagahama Institute of Bio-Science and Technology), Michael Feig (Michigan State University), Tadashi Ando (RIKEN), Koichi Tamura (RIKEN), Takashi Imai (RIKEN), Raimondas Galvelis (RIKEN), Ryuhei Harada (RIKEN), Yasuaki Komuro (RIKEN), Wataru Nishima (RIKEN), Naoyuki Miyashita (RIKEN), Yasuhito Karino (RIKEN)

Acknowledgments: Norio Takase (Isogo Soft), Yasumasa Joti (RIKEN SPring8), Akira Naruse (NVIDIA), Yukihiko Hirano (NVIDIA Japan), Hikaru Inoue (Fujitsu Ltd.), Tomoyuki Noda (Fujitsu Ltd.), Kiyotaka Sakamoto (Fujitsu Ltd.), Yoshinobu Akinaga (VINAS), Yoshitake Sakae (RIST).

GENESIS website

https://www.r-ccs.riken.jp/labs/cbrt/

https://github.com/genesis-release-r-ccs/genesis-2.0-beta-sc20

https://zenodo.org/badge/latestdoi/268506012

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 Notices

> GENESIS is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.
>
> You should have received a copy of the GNU Lesser General Public License along with GENESIS – see the file COPYING and COPYING.LESSER. If not, see https://www.gnu.org/licenses/.

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.

> Copyright ©2006,2007 Mutsuo Saito, Makoto Matsumoto and Hiroshima University. Copyright ©2012 Mutsuo Saito, Makoto Matsumoto, Hiroshima University and The University of Tokyo. All rights reserved.
>
> Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
>
> * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
>
> * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
>
> * Neither the names of Hiroshima University, The University of Tokyo nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
>
> THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

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).

Complementary error function: erfc04

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

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 calculation. Another motivation is to develop a simple MD program, 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 have developed two MD programs in **GENESIS**, namely **SPDYN** (*Sp*atial decomposition *dyn*amics) and **ATDYN** (*At*omic decomposition *dyn*amics).Both **SPDYN** and **ATDYN** are available in **GENESIS1.5**, but we only considered **SPDYN** in **GENESIS2.0.0beta**.

In **SPDYN**, the spatial decomposition scheme is implemented with new parallel algorithms *[11] [12]* and GPGPU calculation.

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

- Atomistic molecular force field (CHARMM, AMBER)
- For extremely large biomolecular systems (more than 10 million atoms), parallel input/output (I/O) scheme is implemented.
- **GENESIS** is optimized for "K computer" and "Fugaku 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 language (90/95/2003) using modules and dynamic memory allocation. No common blocks are used.
- **GENESIS** is free software under the Lesser General Public License version 3 (LGPL). We allow users to use/modify **GENESIS** and redistribute the modified version under the same license.

This user manual mainly provides detailed description of keywords used in the control file. Tutorials for standard MD simulations, REMD simulations, and some analyses are available online (https://www.r-

ccs.riken.jp/labs/cbrt/). We recommend new users of **GENESIS** to start from the next chapter to learn a basic 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 fix 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 the users to check the 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

2.1 GENESIS simulator

Unlike **GENESIS 1.5**, **GENESIS 2.0beta** only supports **SPDYN**. To use **ATDYN** or **Analysis**, please use **GENESIS 1.5**.

2.2 Installation of GENESIS

2.2.1 Requirements

Compilers

GENESIS works on various systems: laptop PCs, workstations, cluster machines, and supercomputers. Since the source code of **GENESIS** is mainly written in Fortran language, Fortran compiler is the first requirement for installation. In addition, "preprocessor" is required, because the source code is "processed" according to the user's computer environment before the compilation. One of the commonly used Fortran compilers is `gfortran`, which is freely available as part of the GNU Compiler Collection (GCC). In this case, `cpp` is selected as a preprocessor, which is also available freely. Another recommended Fortran compiler is `ifort` provided by Intel Corporation which enables us to run the program much faster on Intel CPU. In the Intel compiler package, `fpp` is provided as a preprocessor. Fujitsu compiler `frtpx`, which also functions as a preprocessor, is suitable for Fugaku. In the current release, we only tested Intel and Fujitsu compilers.

MPI and OpenMP

GENESIS work on multiple CPU cores using MPI (Message Passing Interface) and OpenMP protocols (hybrid MPI+OpenMP). MPI and OpenMP are commonly used for parallel computing. In general, MPI is employed for communication between different machines, nodes, or processors, where the memory is not shared among them (distributed-memory). On the other hand, OpenMP is employed in a single processor, and thus, memory is shared in the parallel computation.

OpenMP is natively supported in most modern Fortran compilers. However, the users may have to install MPI libraries by themselves, especially, in the case of laptop PCs and workstations. One of the commonly used MPI software is OpenMPI (https://www.open-mpi.org/). When the users install the OpenMPI libraries in the computer, the users must specify Fortran and C compilers (e.g., `gfortran` and `gcc`) to be used with MPI. After installing the libraries, the users can use `mpif90`, `mpicc`, and

`mpirun`, which are necessary to compile and run the program that is parallelized with MPI. OpenMPI is available freely, and the example installation scheme is shown in *Appendix*. Intel and Fujitsu Corporations are also providing their own MPI libraries for parallel computation.

Mathematical libraries

GENESIS utilizes mathematical libraries such as LAPACK and BLAS (http://www.netlib.org/lapack/). These libraries enable us to efficiently solve complicated mathematical equations such as eigenvalue problems and singular value decomposition. The users have to install these libraries by themselves, if they are not installed in the computer (see *Appendix*). In the case of the Intel and Fujitsu compilers, Intel MKL and Fujitsu SSL II are automatically selected, respectively.

GPGPU

SPDYN works not only with CPU but also with CPU+GPU. Some of the source code in **SPDYN** are written in CUDA, which enables us to effectively run the program on NVIDIA GPU cards. If the users want to run **SPDYN** with GPGPU calculations, the CUDA toolkit (https://developer.nvidia.com/cuda-toolkit) must be also installed in the computer. Note that OpenACC is not employed in **GENESIS** currently.

The recommended compilers, preprocessors, and libraries for **GENESIS** are listed below. Please make sure that at least one of them in each section is installed on your system (GPU is optional). If the users do not use the Intel or Fujitsu compilers, the combination of GCC compiler, GCC preprocessor, and OpenMPI is recommended.

- Operating systems (see *Appendix*)
  - Linux
  - macOS
- Fortran and C compilers
  - GCC compiler `gfortran`, `gcc` (version 4.4.7 or higher is required)
  - Intel compiler `ifort`, `icc`
  - Fujitsu compiler `frtpx`, `fccpx`
- Preprocessors
  - GCC preprocessor `cpp`
  - Intel preprocessor `fpp`
  - Fujitsu compiler `frtpx`
- MPI libraries for parallel computing
  - OpenMPI `mpirun`, `mpif90`, `mpicc`
  - Intel MPI `mpirun`, `mpiifort`, `mpiicc`
  - Fujitsu MPI
- Numerical libraries for mathematical algorithms
  - LAPACK/BLAS

  - Intel Math Kernel Library (MKL)
  - Fujitsu Scientific Subroutine Library (SSL II)
- GPU **(Optional)**
  - NVIDIA GPU cards which support Compute Capability (CC) 6.5 or higher
  - The following GPU cards and CUDA versions have been tested by the GENESIS developers
    * NVIDIA K20, K40, P100, TITAN V, GTX 1080, GTX 1080Ti, RTX 2080, RTX 2080Ti
    * CUDA ver. 8.0, 9.0, 9.1, 9.2, 10.0

Note: If you are using a supercomputer in universities or research institutes, there is a high chance that the system already provides the above requirements so that you don't need to install by yourself. Please refer to the users' guide of the supercomputer, or consult the system administrator.

In general, the latest version of CUDA does not support the latest version of GCC compiler. If you cannot compile GENESIS with CUDA and new GCC compiler, please first make an attempt to install CUDA with older GCC compilers, and then install GENESIS with those CUDA and GCC compilers.

2.2.2 General scheme for installation

Step1. Download the source code

The source code of **GENESIS** is available in the github page of GENESIS2.0beta (https://github.com/genesis-release-r-ccs/genesis-2.0-beta-sc20/). The users first have to uncompress the download file in a appropriate directory. Here, we assume that the users install **GENESIS** in "$HOME/genesis". The "src" directory contains the source code, and general information is written in README.intro.

```
mkdir $HOME/genesis
$ cd $HOME/genesis
$ mv ~Downloads/genesis-2.0-beta-sc20-master.zip ./
$ unzip /genesis-2.0-beta-sc20-master.zip
$ cd genesis-2.0-beta-sc20-master
$ ls
LICENSE         README.md       cleanup         depcomp         version_scripts
Makefile.am     aclocal.m4      compile         fortdep.py
Makefile.in     autom4te.cache  config.status   install-sh
README.howto    bin             configure       missing
README.intro    bootstrap       configure.ac    src
```

Step2. Configure

In order to compile the source code, the users first have to try "autoreconf" or "./bootstrap" command and execute the "configure" script in the directory. This script automatically detects appropriate compilers, preprocessors, and libraries in the users' computer, and create "`Makefile`".

```
$ autoreconf
$ ./configure
```

If you encountered a failure in the configure command, please check the error message carefully. You may have to add appropriate options in this command according to your computer environment (see *Advanced installation*). The followings are possible suggestions to solve frequent problems. Other solutions might be found in the online page (https://www.r-ccs.riken.jp/labs/cbrt/installation/).

- If the error message is like "configure: error: Fortran compiler cannot create executables", Fortran compilers are not detected. The configure script looks for "mpiifort", "mpif90", "mpifrtpx", or "mpifrt" for Fortran compiler, and "mpiicc", "mpicc", "mpifccpx", or "mpifcc" for C compiler. If none of them were detected in your computer, the configure will fail with such error. To solve such problems, you need to set the path to the executables and libraries for MPI in "~/.bashrc" (see *Appendix*) or specify the compilers explicitly in the configure command (see *Advanced installation*). You should also check typing mistakes in the path.
- If the recommended softwares are not used for compilation, warning messages might be displayed in the terminal when the configure command is executed. Those messages are just a warning (not an error), and you may continue the compilation. However, we strongly recommended you to verity the installation in such cases (see *Verify the installation*).
- In some supercomputer systems, "module load [module]" command is required to use compliers, and need to be set before the configure. See the user guide of the system.

Step3. Make install

After the "configure" command is successfully done, type the following command to compile and install **GENESIS**. All **SPDYN** program is compiled and installed into the "./bin" directory by default.

```
$ make install
```

If you encountered a failure, please check the error message carefully. In many cases, errors are caused by invalid path of compilers and libraries. The followings are possible suggestions to solve frequent problems. Other solutions might be found in the online page (https://www.r-ccs.riken.jp/labs/cbrt/installation/).

- If the error message is like "/usr/bin/ld: cannot find -lblas" or "/usr/bin/ld: cannot find -llapack", make sure that the BLAS or LAPACK libraries are installed in the computer (see also *Appendix*). The users may also have to set the path to the libraries in the "configure" command with the "`LAPACK_LIBS`" or "`LAPACK_PATH`" option (see *Advanced installation*).
- If you have installed additional software or libraries to solve a make error, please execute "make clean", and try Step2 and "make install" again.

Step4. Confirmation

After the installation is successfully finished, "spdyn" binary file is found in the "bin" directory.

2.2.3 Advanced installation

In the above scheme, **GENESIS** is installed with default options, and all installed programs run on CPU with double precision calculation. The users can specify additional options in the configure command according to the users' computer environment or desired conditions. The full lists of the available options are obtained by "`./configure --help`". The representative options are as follows.

`--enable-single`

Turn on single precision calculation for force calculations and integrations. Even with this option, constraints and energy values are described by double precision to keep accuracy.

`--enable-mixed`

Turn on single precision calculation for force calculations and double precision calculation for integrations.

`--enable-gpu`

Turn on GPGPU calculation.

`--with-cuda=PATH`

Define path to the CUDA libraries manually.

`--disable-openmp`

Turn off OpenMP parallelization.

`--enable-parallel_IO`

Install the parallel I/O tool (**prst_setup**)

`--enable-debug`

Turn on program debugging (see below)

`--prefix=PREFIX`

Install the programs in the directory designated by `PREFIX`

`--host=Fugaku`

This option enables us to use GENESIS on Fugaku supercomputer. With this, we have the following options:

`--enable-pktimer`

Precise timers for each compoent. By appending `--with-fj_timer_detail`, we can check computational times in more detail. To obtain profiler information (SIMD rate, peak performance, operation number and so on), `--with-fapp` can be written. More detailed information is written in README.howto.

Configuration with non-default compilers

We assume INTEL Fortran/C compilers ("mpiifort" and "mpiicc") by default. Different Fortran compiler can be specified with `FC` and `CC` for Fortran and C compilers, respectively.. For example, in the case of "mpif90" and "mpicc", the following options are added:

```
$ ./configure CC=mpicc FC=mpif90
```

Configuration with an explicit path to LAPACK/BLAS libraries

The following is an example command to set the path to LAPACK and BLAS libraries that are installed in $HOME/Software/lapack-3.8.0/ (see also *Appendix*). Please be careful about the filename of the installed libraries. If the BLAS libraries are installed as "librefblas.a", the option "-lrefblas" must be used. If "librefblas.a" is renamed to "libblas.a", the following command can be used. Linking with the reverse order of "-llapack" and "-lblas" might also cause a failure of installation of **GENESIS**.

```
$ ./configure LAPACK_LIBS="-L$HOME/Software/lapack-3.8.0 -llapack -lblas"
```

or use the "LAPACK_PATH" option:

```
$ ./configure LAPACK_PATH=$HOME/Software/lapack-3.8.0
```

Configuration for GPGPU calculation

In the following command, the users install **SPDYN** that works on CPU+GPU with single-precision calculation. If "--enable-single" is omitted in the command, **SPDYN** works on CPU+GPU with double-precision calculation.

```
$ ./configure --enable-single --enable-gpu
```

Here, if the users encountered an error message like "nvcc: command not found", make sure that the CUDA Toolkit is installed in the computer. In typical Linux workstations or cluster machines, CUDA is installed in "/usr/local/cuda-x.x/" or "/usr/lib/x86_64-linux-gnu/", and "nvcc" should be in a "bin" directory of the install directory. The path to "nvcc" and CUDA libraries should be set in a startup file such as "~/.bashrc". For example, add the following information to "~/.bashrc" in the case of CUDA 9.0,

```
CUDAROOT=/usr/local/cuda-9.0
export PATH=$CUDAROOT/bin:$PATH
export LD_LIBRARY_PATH=$CUDAROOT/lib64:/lib:$LD_LIBRARY_PATH
```

then reload "~/.bashrc" and try the configure command again. If there still remain some troubles, explicitly specify a path to CUDA libraries in the configure command by:

```
$ ./configure --enable-single --enable-gpu --with-cuda=/usr/local/cuda-9.0
```

Configuration for program debugging

If the users encountered memory leak errors during the simulation using **GENESIS**, the origin of the error might be tracked by using a program compiled with a debug option. Note that the debug option makes the calculation much slow. In this case, the runtime check is activated only for CPU codes, even if the "–enable-gpu" option is added to the command.

```
$ ./configure --enable-debug=3
```

Note that `--enable-debug` corresponds to `--enable-debug=1`.

- 0 = no debugging (default)
- 1 = debugging without intensive optimization
- 2 = LEVEL1 + debug information (`-g` and `-DDEBUG`)
- 3 = LEVEL2 + memory check (if possible)
- 4 = LEVEL3 + full check (intel compiler only)

Installation using multiple CPU cores (parallel compile)

In Step3, `-j` option is available, which enables quick compilation of the program using multiple CPU cores. The following command uses 4 CPU cores.

```
$ make -j4 install
```

If you encountered an error message like "Fatal Error: Can't delete temporary module file '...': No such file or directory", please try "make install" without the "`-j`" option.

2.2.4 Verify the installation

The users can verify the installation of **GENESIS** by using test sets which are available in the **GENESIS** webpage (https://www.r-ccs.riken.jp/labs/cbrt/dataset-of-genesis-on-fugaku/). Please, uncompress the download file in an appropriate directory, and move to the "regression_test" directory.

```
$ cd $HOME/genesis
$ mv ~/Downloads/tests_2.0.0.tar.gz ./
$ tar xvf tests_2.0.0.tar.gz
$ cd tests_2.0.0/regression_test
$ ls
build        genesis.pyc   test_common          test_remd
charmm.py    param         test_nonstrict.py    test_spdyn
genesis.py   test.py       test_parallel_IO
```

In the sub-directories in "regression_test", the users can find a lot of input files ("`inp`"), in which various combinations of simulation parameters are specified. In addition, each sub-directory contains output file ("`ref`") obtained by the developers. The users run "test.py" which enables automatic comparison between the users' and developers' results for each MD algorithm.

Run the basic tests

The following is an example command to verify **GENESIS** for basic MD. Here, the programs are executed using 8 CPU core with the "mpirun" command. The users can increase the number of MPI processors according to the users' computer environment, but only 8 are allowed in these tests. Other MPI launchers such as "mpiexec" are also available in the command. There are about 50 test sets, and each test should finish in a few seconds.

```
$ export OMP_NUM_THREADS=1
$ python2 ./test.py "mpirun -np 8 ~/genesis/genesis-2.0-beta-sc20-master/
↪bin/spdyn"
```

If any tests cannot run, please check the following points:

- Number of OpenMP threads should be specified before running the tests.
- Original executable file name (e.g., **spdyn**) must not be changed.
- Python ver. 3 does not work. Please use ver. 2.x, and run "`python2.x ./test.py ...`"
- Regression tests via a queuing system or batch script may not work.

The "test.py" script compares energy in log file between the developer's and user's ones. If the energy differences are less than the tolerance (default = 1.00e-04 for energy), "Passed" is displayed for each test. Among the physical quantities in the log file, virial is the most sensitive to numerical factors, and thus, the tolerance for virial is set to a larger value (1.00e-02). After all tests are finished, the total number of succeeded, failed, and aborted runs will be displayed at the end.

```
Passed  53 / 53
Failed  0 / 53
Aborted 0 / 53
```

If all tests were passed, it means that your **GENESIS** can generate identical results to the developer's **GENESIS**. Note that the developer's **GENESIS** was compiled with Intel compilers, Intel MKL, Open-

MPI library, and the double precision option on Intel CPUs. If your computer system is significantly different from the developer's one, unexpected numerical errors may happen, which can cause failures in some tests. If there were any aborted tests, the users had better to check their log or error files carefully, which exist in the tested sub-directory, and figure out why the error happened. The followings are suggestions to solve typical problems:

- If some tests were aborted due to "memory allocation error", the reason might come from limitation of the memory size. Namely, those tested systems were too large for your computer. The problem should not be so serious.
- Available number of MPI slots in your computer might be actually smaller than the given number of MPI processors. Try to use less number of MPI processors.
- Try to specify the "absolute path" to the program instead of using "relative path".
- Make sure that the MPI environment is properly set.
- Detailed solutions in specific supercomputer systems might be found in the GENESIS website (https://www.r-ccs.riken.jp/labs/cbrt/usage/).

2.2.5 Clean install and re-compilation

The following commands are used to fully recompile **GENESIS**. Note that the direct "make clean" command may not work in the case where `Makefiles` were created in another machine. In this case, the users must run the "./configure" command before "make clean".

```
$ make clean
$ make distclean
$ ./configure [option]
$ make install
```

2.2.6 Uninstall

The user can uninstall **GENESIS** by just removing the program directory. If the user changed the install directory by specifying "`--prefix=PREFIX`" in the configure command, please remove the programs (**atdyn**, **spdyn**, and so on) in the "`PREFIX`" directory.

```
$ rm -rf $HOME/genesis/genesis-2.0-beta-sc20-master
```

2.3 Basic usage of GENESIS

2.3.1 Running GENESIS on a command line

The **GENESIS** programs are executed on a command line. The first argument is basically interpreted as an input file of the program. The input file, which we call *control file* hereafter, contains parameters for simulations. The following examples show typical usage of the **GENESIS** programs. In the case of serial execution,

```
$ [program_name] [control_file]
```

In the case of parallel execution with "mpirun",

```
$ mpirun -np n [program_name] [control_file]
```

For example, **SPDYN** is executed in the following way using 8 MPI processors:

```
$ mpirun -np 8 ~/genesis/genesis-2.0-beta-sc20-master/bin/spdyn INP
```

The users should specify an OpenMP thread number explicitly before running the program. Appropriate number of CPU cores must be used according to the user's computer environment (see also *Available Programs*). For example, if the users want to use 32 CPU cores in the calculation, the following command might be executed.

```
$ export OMP_NUM_THREADS=4
$ mpirun -np 8 ~/genesis/genesis-2.0-beta-sc20-master/bin/spdyn INP
```

2.3.2 Automatic generation of a template control file

Basic usage of each program is shown by executing the program with the `-h` option. In addition, sample control file of each program can be obtained with the `-h ctrl` option:

```
# Show the usage of the program
$ [program_name] -h

# Display a template control file
$ [program_name] -h ctrl [module_name]
```

For example, in the case of **SPDYN**, the following messages are displayed:

```
$ spdyn -h

# normal usage
  % mpirun -np XX ./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

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

(skipped...)
```

This message tells the users that **SPDYN** can be executed with mpirun. A template control file for molecular dynamics simulation (md) can be generated by executing **SPDYN** with the `-h ctrl md` option. The same way is applicable for energy minimization (min), replica exchange simulation (remd), and replica path sampling simulation (rpath). The template control file for energy minimization is shown below. If the users want to show all available options, please specify `ctrl_all` instead of `ctrl`. The users can edit this template control file to perform the simulation that the users want to do.

```
$ ~/genesis/genesis-2.0-beta-sc20-master/bin/spdyn -h ctrl min > INPMIN

$ less INPMIN

[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    # [CHARMM,AMBER,GROAMBER,GROMARTINI]
electrostatic = PME       # [CUTOFF,PME]
switchdist    = 10.0      # switch distance
cutoffdist    = 12.0      # cutoff distance
pairlistdist  = 13.5      # pair-list distance

[MINIMIZE]
method        = SD        # [SD]
nsteps        = 100       # number of minimization steps

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

2.4 Control file

In the control file, detailed simulation conditions are specified. The control file consists of several sections (e.g., **[INPUT]**, **[OUTPUT]**, **[ENERGY]**, **[ENSEMBLE]**, and so on), each of which contains closely-related keywords. For example, in the **[ENERGY]** section, parameters are specified for the potential energy calculation such as a force field type and cut-off distance. In the **[ENSEMBLE]** section, there are parameters to select the algorithm to control the temperature and pressure in addition to the target temperature and pressure of the system. Here, we show example control files for the energy minimization and normal molecular dynamics simulations.

2.4.1 Example control file for the energy minimization

The control file for the energy minimization must include a **[MINIMIZE]** section (see *Minimize section*). By using the following control file, the users carry out 2,000-step energy minimization with the steepest descent algorithm (SD). The CHARMM36m force field is used, and the particle mesh Ewald (PME) method is employed for the calculation of long-range interaction.

```
[INPUT]
topfile = top_all36_prot.rtf      # topology file
parfile = par_all36m_prot.prm     # parameter file
strfile = toppar_water_ions.str   # stream file
psffile = build.psf               # protein structure file
pdbfile = build.pdb               # PDB file

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

[ENERGY]
forcefield        = CHARMM        # CHARMM force field
electrostatic     = PME           # Particl mesh Ewald method
switchdist        = 10.0          # switch distance (Ang)
cutoffdist        = 12.0          # cutoff distance (Ang)
pairlistdist      = 13.5          # pair-list cutoff distance (Ang)
pme_nspline       = 4             # order of B-spline in PME
pme_max_spacing   = 1.2           # max grid spacing allowed (Ang)
vdw_force_switch  = YES           # turn on van der Waals force switch
contact_check     = YES           # turn on clash checker

[MINIMIZE]
method            = SD            # Steepest descent method
nsteps            = 2000          # number of steps
eneout_period     =  100          # energy output freq
crdout_period     =  100          # coordinates output frequency
rstout_period     = 2000          # restart output frequency
nbpdate_period    =   10          # pairlist update frequency

[BOUNDARY]
type              = PBC           # periodic boundary condition
box_size_x        = 64.0          # Box size in X dimension (Ang)
box_size_y        = 64.0          # Box size in Y dimension (Ang)
box_size_z        = 64.0          # Box size in Z dimension (Ang)
```

2.4.2 Example control file for normal MD simulations

The control file for normal MD simulations must include a **[DYNAMICS]** section (see *Dynamics section*). By using the following control file, the users carry out a 100-ps MD simulation at T = 298.15 K and P = 1 atm in the NPT ensemble. The equations of motion are integrated by the RESPA algorithm with a time step of 2.5 fs, and the bonds of light atoms (hydrogen atoms) are constrained using the SHAKE/RATTLE and SETTLE algorithms. The temperature and pressure are controlled with the Bussi thermostat and barostat.

```
[INPUT]
topfile = top_all36_prot.rtf      # topology file
parfile = par_all36m_prot.prm     # parameter file
strfile = toppar_water_ions.str   # stream file
psffile = build.psf               # protein structure file
pdbfile = build.pdb               # PDB file
rstfile = min.rst                 # restart file

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

[ENERGY]
forcefield        = CHARMM        # CHARMM force field
electrostatic     = PME           # Particl mesh Ewald method
switchdist        = 10.0          # switch distance (Ang)
cutoffdist        = 12.0          # cutoff distance (Ang)
pairlistdist      = 13.5          # pair-list cutoff distance (Ang)
pme_nspline       = 4             # order of B-spline in PME
pme_max_spacing   = 1.2           # max grid spacing allowed (Ang)
vdw_force_switch  = YES           # turn on van der Waals force switch

[DYNAMICS]
integrator        =   VRES        # RESPA integrator
timestep          = 0.0025        # timestep (2.5fs)
nsteps            =  40000        # number of MD steps (100ps)
eneout_period     =    400        # energy output period (1ps)
crdout_period     =    400        # coordinates output period (1ps)
rstout_period     =  40000        # restart output period
nbupdate_period   =     10        # nonbond update period
elec_long_period  =      2        # period of reciprocal space calculation
thermostat_period =     10        # period of thermostat update
barostat_period   =     10        # period of barostat update

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

[ENSEMBLE]
ensemble          = NPT           # NPT ensemble
tpcontrol         = BUSSI         # BUSSI thermostat and barostat
temperature       = 300           # target temperature (K)
pressure          = 1.0           # target pressure (atm)

[BOUNDARY]
type              = PBC           # periodic boundary condition
```

CHAPTER

THREE

AVAILABLE PROGRAMS

3.1 Simulators

3.1.1 Basic functions of SPDYN

SPDYN is parallelized with the domain decomposition scheme. The program is designed to achieve high-performance molecular dynamics simulations, such as micro-second simulations and cellular-scale simulations. The program runs on not only CPU but also CPU+GPU with the hybrid MPI+OpenMP protocol. Here, beside double-precision, mixed-precision or single-precision calculations are also available.

3.1.2 Hybrid MPI+OpenMP calculation in SPDYN

The users had better to understand a basic scheme of parallel calculation in **SPDYN** to get the best performance in the calculation. As described above, the simulation box is divided into domains according to the number of MPI processors. Each domain is further divided into smaller cells, each of whose size is adjusted to be approximately equal to or larger than the half of "pairlistdist + α". Here, "pairlistdist" is specified in the control file, and α depends on the algorithms used in the simulation (see the next subsection). Note that all domains or cells have the same size with a rectangular or cubic shape. Each MPI processor is assigned to each domain, and data transfer or communication about atomic coordinates and forces is achieved between only neighboring domains. In addition, calculation of bonded and non-bonded interactions in each domain is parallelized based on the OpenMP protocol. These schemes realize hybrid MPI+OpenMP calculation, which is more efficient than flat MPI calculation on recent computers with multiple CPU cores. Because MPI and OpenMP are designed for distributed-memory and shared-memory architectures, respectively, MPI is mainly used for parallelization between nodes and OpenMP is used within one node.

The following figures illustrate how the hybrid MPI+OpenMP calculations are achieved in **SPDYN**. In Fig. 3.1 (a) and (b), 8 MPI processors with 4 OpenMP threads (32 CPU cores in total), and 27 MPI processors with 2 OpenMP threads (54 CPU cores in total) are used, respectively. In these Figures, only XY dimensions are shown for simplicity.

For Case (a), the following commands are used:

```
$ export OMP_NUM_THREADS=4
$ mpirun -np 8 ~/genesis/genesis-2.0-beta-sc20-master/bin/spdyn INP > log
```

For Case (b), the following commands are used:

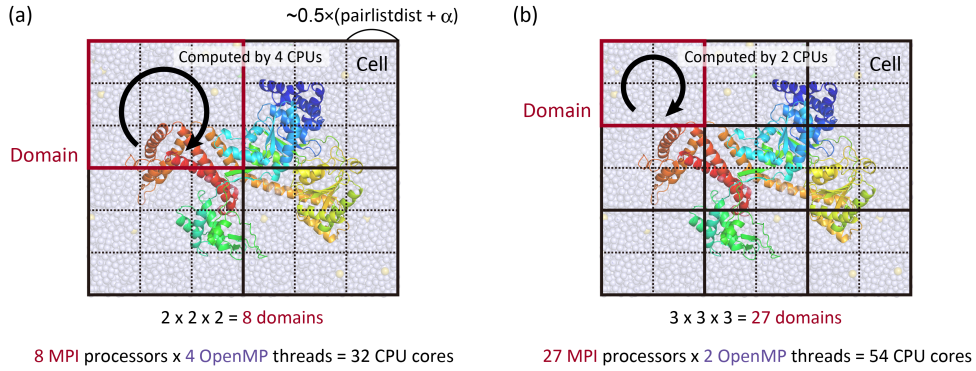


Fig. 3.1: Schematic representation of the hybrid MPI+OpenMP calculation in SPDYN.

```
$ export OMP_NUM_THREADS=2
$ mpirun -np 27 ~/genesis/genesis-2.0-beta-sc20-master/bin/spdyn INP > log
```

In the log file, the users can check whether the given numbers of MPI processors and OpenMP threads are actually employed or not. The following information should be found in the log file for instance for Case (a):

```
[STEP2] Setup MPI

Setup_Mpi_Md> Summary of Setup MPI
  number of MPI processes    =          8
  number of OpenMP threads   =          4
  total number of CPU cores  =         32
```

Note: In most cases, the number of domains in each dimension is automatically determined according to the given number of MPI processors. However, if such automatic determination is failed, the users must specify the number of domains explicitly in the control file (see *Boundary section*).

3.1.3 Limitation of the available MPI processors

Basically, there are a few limitations in **SPDYN**. First, the number of domains must be equal to the number of MPI processors. Second, one domain must be composed of at least 8 cells (= 2×2×2), where the cell size in one dimension is automatically set to be larger than the half of "pairlistdist + 2.0 + α" (2.0 should be added for assigining CHn groups in the cell cell for)., The following table summarizes the default α value in each algorithm. It can be also defined by the user (see *Boundary section*). According to these rules, the available "maximum" number of MPI processors ($N_{\max}$) for a certain target system is mainly determined by the simulation box size and "pairlistdist". For example, if the box size of your target system is $64\times64\times64\text{Å}^3$, and "pairlistdist=13.5" is specified in the control file, $N_{\max}$ is $4\times4\times4 = 64$ in the case of NVT ensemble. If the users want to use much more CPU cores than $N_{\max}$, the number of OpenMP threads should be increased instead of the MPI processors.

In the MD simulation with the NPT ensemble, these rules become more important, because the box size (or cell size) can change during the simulation. For accurate energy evaluation, the cell size should be greater than the half of "pairlistdist + 2.0". If the box size is decreased during the simulation, cell size could be less than half of "pairlistdist + 2.0" and the simulation result will not be reliable any more.

α greater than zero is assigned to avoid such a problem. If the users encountered the following error message in the simulation, the problem is probably related to the above rules, where the specified number of MPI processors might exceed $N_{\max}$.

```
Setup_Processor_Number> Cannot define domains and cells. Smaller or
adjusted MPI processors, or shorter pairlistdist, or larger boxsize
should be used.
```

In this case, please make sure that one domain can be composed of at least 2 cells in each dimension. If the domains and cells are successfully determined, they can be seen in the early part of the log file. The following example is corresponding to the situation in Fig. 3.1 (b).

```
Setup_Boundary_Cell> Set Variables For Boundary Condition
  domains (x,y,z) =          3          3          3
  ncells (x,y,z)  =          6          6          6
```

3.1.4 Available sections

Fundamental functions in **SPDYN** are energy minimization (Min), molecular dynamics method (MD), replica-exchange method (REMD), and string method (String). As shown in the last part of the previous chapter, the users carry out simulations of these methods by writing related sections in the control file. The users can extend these fundamental functions by combining various sections. For example, to run a "restrained MD simulation", the users add **[SELECTION]** and **[RESTRAINTS]** sections in the control file of the "normal MD simulation".

3.2 Parallel I/O tools

SPDYN can be employed with the parallel I/O protocol to achieve massively parallel computation. Since **SPDYN** is parallelized with the domain decomposition scheme, each MPI processor has the coordinates of atoms in the assigned domain. Therefore, large ammount of communication is needed between MPI processors to write the coordinates in a single DCD file, which is a waste of time in the case of the simulations for a huge system like 100,000,000 atoms. To avoid such situations, file I/O in each node (parallel I/O) is useful. The following tools are used to handle the files generated from parallel I/O simulations.

prst_setup

> This tool divides input files (PDB and PSF) for a huge system into multiple files, where each file is assinged to each domain. The obtained files can be read as restart files in the **[INPUT]** section. Note that **prst_setup** is not compiled with Fujitsu compilers. Therefore, if the users are going to perform MD simulations with parallel I/O in Fujitsu supercomputers, the users must create the files without using Fujitsu compilers elsewhere in advance. Even if **prst_setup** and **SPDYN** are compiled with different compilers, there is no problem to execute **SPDYN** with parallel I/O.

CHAPTER

FOUR

INPUT SECTION

4.1 How to prepare input files

In order to run MD simulations, the users have to prepare input files that contains information about the coordinates of the initial structure as well as topology of the system and force field parameters. The users first create those input files by using a setup tool, and their filenames are specified in the **[INPUT]** section of the control file. **GENESIS** supports various input file formats such as CHARMM, AMBER, and GROMACS. Basically, required input files depend on the force field to be used in the simulation. The following table summarizes the essential input files and setup tools for each force field.

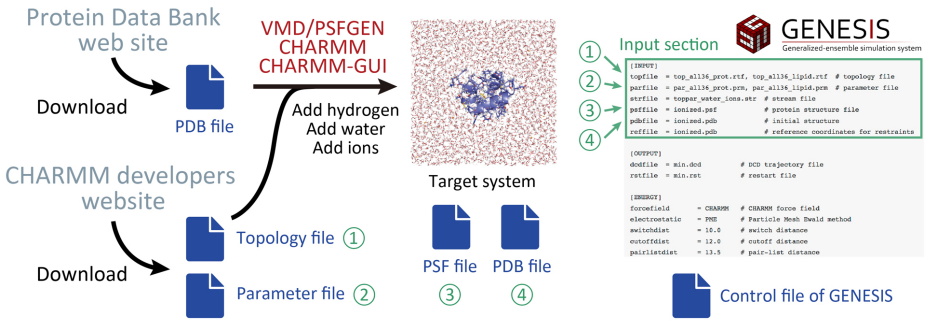
Force field	Input files	Setup tool
CHARMM	top, par, psf, pdb (or crd), str	VMD, PSFGEN, CHARMM-GUI, CHARMM
AMBER	prmtop, pdb, (or ambcrd)	LEaP

4.1.1 CHARMM force field

One of the commonly used parameters for biomolecules is the CHARMM force field, which was originally developed by the Karplus group at the Harvard University *[7]*. The users can obtain the files that contain the force field parameters from the CHARMM group's web site. At this momemt, the latest version of the CHARMM force field is C36m *[13]*. In the download file, there are topology and parameter files (e.g., top_all36_prot.rtf and par_all36m_prot.prm).

In order to run the MD simulation with the CHARMM force field, the users have to additionally make a new file that holds the information about the atom connectivity of the "whole" target system. Note that the topology file (e.g., top_all36_prot.rtf) does not contain such information, because it is designed to generally define the topology of proteins by dealing with the 20 amino acid residues as "fragments". In order to hold the topology information of the target system, the users will create a "PSF" file (protein structure file). It is commonly used in other MD software, and can be generated from the PDB and topology files by using VMD/PSFGEN *[14]*, CHARMM-GUI *[15]*, or the CHARMM program *[2]*.

When the PSF file is created, "processed PDB" file is also obtained, where the atom name or residue name might be changed from those in the original PDB file. The users must use this PDB file as the input of the MD simulation, because it has a consistency with the information in PSF. Consequently, the users need four files (processed PDB, parameter, topology, and PSF) as the inputs of **GENESIS**. These files are specified in the [INPUT] section of the control file of **GENESIS**.



4.1.2 AMBER force field

The AMBER force field has been also commonly used for the MD simulations of biomolecules, which was originally developed by the Kollman group at the University of California, San Francisco *[16]*. **GENESIS** can deal with the AMBER force fields. Basic scheme to prepare the input files for **GENESIS** is similar to that in the case of CHARMM. The users utilize the LEaP program in AmberTools *[1]*. LEaP has a similar function to PSFGEN. After building the target system using LEaP, the users obtain PRMTOP, CRD, and PDB files. PRMTOP contains the information about parameter and topology of the target system, and CRD and PDB include the coordinates of atoms in the initial structure. **GENESIS** uses these files as the inputs.

4.2 General input files

topfile

CHARMM topology file containing information about atom connectivity of residues and other molecules. For details on the format, see the CHARMM web site *[25]*.

parfile

CHARMM parameter file containing force field parameters, e.g. force constants and equilibrium geometries.

strfile

CHARMM stream file containing both topology information and parameters.

psffile

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

prmtopfile

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 *[26]*.

grotopfile

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 *[27]*.

pdbfile

Coordinates file in the PDB format. If *rstfile* is also specified in the **[INPUT]** section, coordinates in *pdbfile* are replaced with those in *rstfile*.

crdfile

Coordinates file in the CHARMM format. If *pdbfile* is also specified in the **[INPUT]** section, coordinates in *crdfile* are NOT used. However, if *pdbfile* is not specified, coordinates in *crdfile* are used. If *rstfile* is further specified, coordinates in *rstfile* are used.

ambcrdfile

Coordinates file in the AMBER format (ascii). If *pdbfile* is also specified in the **[INPUT]** section, coordinates in *ambcrdfile* are NOT used. However, if *pdbfile* is not specified, coordinates in *ambcrdfile* are used. If *rstfile* is further specified, coordinates in *rstfile* are used.

grocrdfile

Coordinates file in the GROMACS format (.gro file). If *pdbfile* is also specified in the **[INPUT]** section, coordinates in *grocrdfile* are NOT used. However, if *pdbfile* is not specified, coordinates in *grocrdfile* are used. If *rstfile* is further specified, coordinates in *rstfile* are used. Note that velocites and simulation box size in *grocrdfile* are NOT used.

rstfile

Restart file in the GENESIS format. This file contains atomic coordinates, velocities, simulation box size, and other variables which are essential to restart the simulation continuously. If *rstfile* is specified in the **[INPUT]** section, coordinates in *pdbfile*, *crdfile*, *grocrdfile*, or *ambcrdfile* are replaced with those in *rstfile*. The box size specified in the **[BOUNDARY]** seciton is also overwritten. Note that *pdbfile*, *crdfile*, *grocrdfile*, or *ambcrdfile* should be still specified in the **[INPUT]** section, even if *rstfile* is specified.

Note that the file format of *rstfile* was changed after ver. 1.1.0. The *rst_upgrade* tool enables us to change the old format used in ver. 1.0.0 or older to the new one.

4.3 Input files for restraint

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*.

ambreffile

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*.

groreffile

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*.

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.

localresfile (for **SPDYN** only)

This file defines restraints to be applied in the system. If you are not an expert of GENESIS, we strongly recommend you to simply use the **[RESTRAINTS]** section for restraint instead of using **localresfile**.

In **localresfile**, only bond, angle, and dihedral angle restraints can be defined. In addition, selected atoms in **localresfile** must exist in the same cell in the domain decomposition scheme. The restraint energy calculated for the lists in **localresfile** is NOT explicitly displayed in the log file. Instead, the local restraint energy is hidden in the conventional bond, angle, and dihedral angle energy terms of the log file.

The restraint potentials defined in **localresfile** are given by 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 syntax in **localresfile** is as follows:

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

The users must carefully specify the atom index in this file. The atom indexes in **localresfile** must be consistent with those in the other input files such as **psffile**.

The following is an example of **localresfile**:

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

4.4 Input files for REMD and RPATH simulations

In the REMD or RPATH simulations, input files (mainly coordinates and restart files) should be prepared for each replica. In GENESIS, we can easily specify those multiple files in the **[INPUT]** section. If we include '{}' in the input filename, {} is automatically replaced with the replica index. For example, in the case of REMD simulations with 4 replicas, we prepare input_1.pdb, input_2.pdb, input_3.pdb, and input_3.pdb, and specify `pdbfile = input_{}.pdb` in the **[INPUT]** section. This rule is also applicable to the restart filename.

fitfile (for RPATH only; GENESIS 1.1.5 or later)

Reference coordinates for structure fitting. This file is only used in the string method. For other cases (MD, MIN, or REMD), *reffile*, *groreffile*, or *ambreffile* is used for reference coordinates for fitting, and this *fitfile* is simply ignored, even if it is specified in the **[INPUT]** section.

4.5 Examples

MD simulations of proteins in explicit solvent with the CHARMM36m force field:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf
parfile = ../toppar/par_all36m_prot.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
```

MD simulations with positional restraint:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf
parfile = ../toppar/par_all36m_prot.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
reffile = ../build/input.pdb
```

MD simulations of membrane proteins with the CHARMM36m force field:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf, ../toppar/top_all36_lipid.rtf
parfile = ../toppar/par_all36m_prot.prm, ../toppar/par_all36_lipid.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
```

In this case, we specify multiple top and par files for proteins and lipids separated by commas.

If one line becomes very long, backslash " \ " can be used as a line continuation character:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf, \
          ../toppar/par_all36_na.prm,   \
          ../toppar/top_all36_lipid.rtf
parfile = ../toppar/par_all36m_prot.prm, \
          ../toppar/top_all36_na.rtf,    \
          ../toppar/par_all36_lipid.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
```

MD simulations with the AMBER force field:

```
[INPUT]
prmtopfile = ../build/input.prmtop
ambcrdfile = ../build/input.crd
```

MD simulations with the all-atom Go-model:

```
[INPUT]
grotopfile = ../build/input.top
grocrdfile = ../build/input.gro
```

In this case, we specify grotop and grocrd files obtained from the SMOG server or SMOG2 software.

REMD simulations starting from the same initial structure:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf
parfile = ../toppar/par_all36m_prot.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
```

REMD simulations starting from different initial structures:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf
parfile = ../toppar/par_all36m_prot.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input_rep{}.pdb
```

REMD simulations with restarting:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf
parfile = ../toppar/par_all36m_prot.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
rstfile = run_rep{}.rst
```

CHAPTER

FIVE

OUTPUT SECTION

GENESIS yields trajectory data (coordinates and velocities) in the *DCD* file format regardless of the force field or MD algorithm. **GENESIS** can also generate a restart file (*rstfile*) during or at the end of the simulation, which can be used to restart and extend the simulation continuously. Output frequency of each file (e.g., crdout_period and velout_period) is specified in the **[DYNAMICS]** section in the case of the MD, REMD, and RPATH simulations, or **[MINIMIZE]** section in the case of the energy minimization.

5.1 General output files

dcdfile

Filename for the coordinates trajectory data. Coordinates are written in the DCD format, which is commonly used in various MD software such as CHARMM and NAMD. The filename must be given in the case of `crdout_period > 0`. However, if `crdout_period = 0` is specified in the control file, no **dcdfile** is generated, even if the filename is specified in the **[OUTPUT]** section.

dcdvelfile

Filename for the velocity trajectory data. Velocities are written in the DCD format. The filename must be given in the case of `velout_period > 0`. However, if `velout_period = 0` is specified in the control file, no **dcdvelfile** is generated, even if the filename is specified in the **[OUTPUT]** section.

rstfile

Filename for the restart data. The rstfile contains coordinates, velocities, simulation box size, and so on. This file can be used to extend the simulation continuously. In addition, it can be used to switch the simulation algorithms (e.g., from minimization to MD, from MD to REMD, from REMD to minimization, etc) The filename must be given in the case of `rstout_period > 0`. However, if `rstout_period = 0` is specified in the control file, no **rstfile** is generated, even if the filename is specified in the **[OUTPUT]** section.

5.2 Output files in REMD and RPATH simulations

When the user performs REMD or RPATH simulations, the user must include '{}' in the output filename. This {} is automatically replaced with the replica index.

remfile (only for REMD simulations)

This file contains parameter index data from the REMD simulation, which is written for each replica every `exchange_period` steps. This is used as an input file for the *remd_convert* tool to sort the coordinates trajectory data by parameters. The filename must contain '{}', which is automatically replaced with the replica index. Note that the information about the parameter index as well as replica index in the entire REMD simulation is written in the standard (single) output file (see online Tutorials).

logfile (only for REMD and RPATH simulations)

This file contains the energy trajectory data from the REMD or RPATH simulations, which is written for each replica every `exchange_period` steps. This is used as an input file for the *remd_convert* tool to sort the coordinates trajectory data by parameters. The filename must contain '{}', which is automatically replaced with the replica index.

rpathfile (only for RPATH simulations)

This file contains the trajectory of image coordinates in the string method, which are reference values used in the restraint functions. Columns correspond to the collective variables, and rows are time steps. This data is written with the same timing as the *dcdfile*. For details, see *RPATH section*.

enefile (only for gREST simulations)

This file contains the energy information for all replicas with temperature parameters. We can obtain free energy based on this file. The file output period is same as dcdfile case. Please note that you should define **Analysis_grest=Yes** in **[REMD]** section.

5.3 Examples

For normal MD simulations:

```
[OUTPUT]
dcdfile = run.dcd
rstfile = run.rst
```

For REMD simulations:

```
[OUTPUT]
logfile = run_rep{}.log
dcdfile = run_rep{}.dcd
remfile = run_rep{}.rem
rstfile = run_rep{}.rst
```

CHAPTER

SIX

ENERGY SECTION

6.1 Force fields

In general, potential energy function is given by:

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

where K_b, K_θ, K_ϕ, and K_{ϕ_i} are the force constant of the bond, angle, dihedral angle, and improper dihedral angle term, respectively, and b_0, θ_0, ϕ_0, and $\phi_{i,0}$ are corresponding equilibrium values. 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 detailed formula and parameters in the potential energy function depend on the force field and molecular model.

---

forcefield *CHARMM / AMBER / GROAMBER / GROMARTINI*

Default : CHARMM

Type of the force field used for energy and force calculation. For the AMBER force field, the scheme used in the GROMACS program package is availble in addition to that used in the AMBER package. In this case, calculation for the dispersion correction term and truncation of the non-bonded energy term are different between AMBER and GROMACS.

- **CHARMM**: CHARMM force field with the all-atom model (CHARMM22, 27, 36, 36m) *[7] [28] [29] [30]*
- **AMBER**: AMBER force field with the original AMBER scheme *[6]*
- **GROAMBER**: AMBER force field with the GROMACS scheme
- **GROMARTINI**: MARTINI model *[18] [31]*

6.2 Non-bonded interactions

Calculation of the non-bonded interaction is the most time consuming part in MD simulations. Computational time for the non-bonded interaction terms without any approximation is proportional to $O(N^2)$. To reduce the computational cost, a cut-off approximation is introduced, where the energy and force calculation is truncated at a given cut-off value (keyword *cutoffdist*). Simple truncation at the cut-off distance 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*), which is generally applied to the van der Waals interactions (keyword *switchdist*). There are two kinds of switching: "potential switch" and "force switch". In **GENESIS**, potential switching is turned on as the default. However, in the case of the AMBER force field, potential switching is still turned off, since the original AMBER program package is not using the potential switching. To turn on the "force switching", `vdw_force_switch=YES` must be specified. Note that the cut-off scheme for the electrostatic energy term is different from that for the van der Waals energy term, where the former uses a shift function. Such shift is turned on when `Electrostatic=Cutoff` is specified.

---

electrostatic *CUTOFF / PME*

Default : PME

- **CUTOFF**: Non-bonded interactions including the van der Waals interaction are just truncated at *cutoffdist*.
- **PME**: Particle mesh Ewald (PME) method is employed for long-range interactions. This option is only availabe in the periodic boundary condition.

switchdist *Real*

Default : 10.0 (unit : Angstrom)

Switch-on distance for nonbonded interaction energy/force quenching. If *switchdist* is set to be equal to *cutoffdist*, switching can be turned off. Switching scheme depends on the selected force field, *vdw_shift*, and *vdw_force_switch* parameters. In the case of AMBER force field, this switching must be disabled, because the switching function is not available. In the case of "forcefield = GROMARTINI" and "electrostatic = CUTOFF", *switchdist* is used only in the van der Waals potential energy. The switching-on distance for the electrostatic energy is automatically defined as 0.0.

cutoffdist *Real*

Default : 12.0 (unit : Angstrom)

Cut-off distance for the non-bonded interactions. This distance must be larger than *switchdist*, while smaller than *pairlistdist*. In the case of the AMBER force field, this value must be equal to *switchdist*.

pairlistdist *Real*

Default : 13.5 (unit : Angstrom)

Distance used to make a Verlet pair list for non-bonded interactions *[32]*. This distance must be larger than *cutoffdist*.

dielec_const *Real*

Default : 1.0

Dielectric constant of the system. Note that the distance dependent dielectric constant is not availabe in **GENESIS**.

vdw_force_switch *YES / NO*

Default : NO

This paramter determines whether the force switch function for van der Waals interactions is employed or not. *[33]* The users must take care about this parameter, when the CHARMM force field is used. Typically, "vdw_force_switch=YES" should be specified in the case of CHARMM36.

vdw_shift *YES / NO*

Default : NO

This parameter determines whether the energy shift for the van der Waals interactions is employed or not. If it is 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".

dispersion_corr *NONE / ENERGY / EPRESS*

Default : NONE (automatically set to **EPRESS** in the case of AMBER)

This parameter determines how to deal with the long-range correction about the cut-off for the van der Waals interactions. Note that the formula used for the correction is different between the GROMACS and AMBER schemes. In the case of the CHARMM force filed, "dispersion_corr=NONE" is always used.

- **NONE**: No correction is carried out.
- **ENERGY**: Only energy correction is carried out.
- **EPRESS**: Both energy and internal pressure corrections are carried out.

contact_check *YES / NO*

Default : NO

If this parameter is set to *YES*, length of all covalent bonds as well as distance between non-bonded atom pairs are checked at the begining of the simulation. If long covalent bonds or clashing atoms are detected, those atom indexes are displayed in the log file. If *contact_check* is turned on, *nonb_limiter* is also automatically enabled. If the users want to turn on only "contact_check", please specify "contact_check = YES" and "nonb_limiter = NO" explicitly. Note that this contact_check does not work in the parallel-io scheme. If you are using **SPDYN**, please see also *structure_check*.

structure_check *NONE / FIRST / DOMAIN*

Default : NONE

If this parameter is set to FIRST or DOMAIN, length of all covalent bonds as well as distance between non-bonded atom pairs are checked at the begininig or during the simulation. This option is similar to *contact_check*, but has an improved capability when the parallel-io scheme is employed. In **SPDYN**, we recommend the users to use this option instead of *contact_check*. Since the structure check spends additional computational time, the users had better to turn off this option in the production run.

- **NONE**: Do not check the structure

- **FIRST**: Check the structure only at the beginning of the simulation
- **DOMAIN**: Check the structure whenever the pairlist is updated

nonb_limiter *YES / NO*

Default : NO (automatically set to be equal to **contact_check**)

If this parameter is set to *YES*, large force caused by the atomic clash is suppressed during the simulation. Here, the atomic clash can be defined by *minimum_contact* (see below). If "contact_check = YES" is specified, this parameter is automatically set to "YES". If the users want to turn on only "contact_check", please specify "contact_check = YES" and "nonb_limiter = NO" explicitly. This option is basically useful for the energy minimization or equilibration of the system. However, we strongly recommend the users to turn off this option in the production run, because suppression of large forces is an "unphysical" manipulation to avoid unstable simulations.

minimum_contact *Real*

Default : 0.5 (unit : Angstrom)

This parameter defines the clash distance, when `contact_check = YES` is specified. If the distance between the non-bonded atoms is less than this value, energy and force are computed using this distance instead of the actual distance.

nonbond_kernel *AUTOSELECT / GENERIC / KGENERIC / FUGAKU / INTELKNL / GPU*

Default: AUTOSELECT

If this parameter is set to AUTOSELECT, the program automatically decide the kernel for the real-space non-bonded interaction. Please remember that you should compile with GPU option to define GPU as nonboned_kernel.

6.3 Particle mesh Ewald method

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 *[34] [35]*, 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.

---

pme_alpha *Real or auto*

Default : auto

Exponent of complementary error function. If `pme_alpha=auto` is specified, the value is automatically determined from *cutoffdist* and *pme_alpha_tol*.

Note: The default of pme_alpha was 0.34 in GENESIS ver. 1.1.0 or former.

pme_alpha_tol *Real*

Default : 1.0e-5

Tolerance to be used for determining *pme_alpha*, when `pme_alpha=auto` is specified.

pme_nspline *Integer*

Default : 4

B-spline interpolation order used for the evaluation of $b_1(G_1)$, $b_2(G_2)$, $b_3(G_3)$, and $\mathbf{Q}$. The order must be $>= 3$.

pme_max_spacing *Real*

Default : 1.2 (unit : Angstrom)

Max PME grid size used in the automatic grid number determination 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*

Default : N/A (Optional)

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

pme_ngrid_y *Integer*

Default : N/A (Optional)

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

pme_ngrid_z *Integer*

Default : N/A (Optional)

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

PME_scheme *AUTOSELECT / OPT_1DALLTOALL / NOOPT_1DALLTOALL / OPT_2DALLTOALL / NOOPT_2DALLTOALL*

Default : AUTOSELECT

AUTOSELECT chooses the best scheme by running in setup procedure. Other schemes can be chosen manually according to your preference.. For *1DALLTOALL* and *2DALLTOALL*, see ref *[36]* for details.

SPDYN use OpenMP/MPI hybrid parallel fast Fourier transformation library, FFTE *[37]*. 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. OPT_1DALLTOALL or NOOPT_1DALLTOALL:

   `pme_ngrid_x` should be multiple of 2 $\times$ `domain_x`.

   `pme_ngrid_y` should be multiple of `domain_y` $\times$ `domain_z` if `domain_z` is an even number and multiple of `domain_y` $\times$ `domain_z` $\times$ 2 otherwise.

   `pme_ngrid_z` should be multiple of `domain_x` $\times$ `domain_z` and multiple of `domain_y` $\times$ `domain_z`.

2. OPT_2DALLTOALL or NOOPT_2DALLTOAlL:

   `pme_ngrid_x` should be multiple of 2 $\times$ `domain_x`.

   `pme_ngrid_y` should be multiple of `domain_y` $\times$ `domain_z` if `domain_z` is an even number and multiple of `domain_y` $\times$ `domain_z` $\times$ 2 otherwise.

   `pme_ngrid_z` should be multiple of `domain_x` $\times$ `domain_z`.

3. NOOPT_1DALLTOALL or NOOPT_2DALLTOALL:

   Cell size in each dimension divided by grid spacing should be greater than `pme_nsplie`.

4. OPT_1DALLTOALL or OPT_2DALLTOALL:

   `pme_ngrid_[x,y,z]` divided by `domain_[x,y,z]` should be greater than $\times$ `pme_nspline`

6.4 Lookup table

The following keywords are relevant if CHARMM or AMBER 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 *[38]* 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".

6.5 Examples

Simulation with the CHARMM36 force field in the periodic boundary condition:

```
[ENERGY]
forcefield       = CHARMM  # CHARMM force field
electrostatic    = PME     # use Particle mesh Ewald method
switchdist       = 10.0    # switch distance
cutoffdist       = 12.0    # cutoff distance
pairlistdist     = 13.5    # pair-list distance
vdw_force_switch = YES     # force switch option for van der Waals
pme_nspline      = 4       # order of B-spline in [PME]
pme_max_spacing  = 1.2     # max grid spacing allowed
```

Simulation with the AMBER force field in the periodic boundary condition:

```
[ENERGY]
forcefield       = AMBER   # AMBER force field
electrostatic    = PME     # use Particle mesh Ewald method
switchdist       = 8.0     # switch distance
cutoffdist       = 8.0     # cutoff distance
pairlistdist     = 9.5     # pair-list distance
pme_nspline      = 4       # order of B-spline in [PME]
pme_max_spacing  = 1.2     # max grid spacing allowed
```

Recommended options in the case of energy minimization (see *Minimize section*) for the initial structure with the CHARMM36 force field:

```
[ENERGY]
forcefield       = CHARMM  # CHARMM force field
electrostatic    = PME     # use Particle mesh Ewald method
switchdist       = 10.0    # switch distance
cutoffdist       = 12.0    # cutoff distance
pairlistdist     = 13.5    # pair-list distance
vdw_force_switch = YES     # force switch option for van der Waals
pme_nspline      = 4       # order of B-spline in [PME]
pme_max_spacing  = 1.2     # max grid spacing allowed
contact_check    = YES     # check atomic clash
nonb_limiter     = YES     # avoid failure due to atomic clash
minimum_contact  = 0.5     # definition of atomic clash distance
```

Simulations with the GB/SA implicit solvent model

```
[ENERGY]
forcefield       = CHARMM  # [CHARMM]
electrostatic    = CUTOFF  # [CUTOFF]
switchdist       = 23.0    # switch distance
cutoffdist       = 25.0    # cutoff distance
pairlistdist     = 27.0    # pair-list distance
implicit_solvent = GBSA    # Turn on GBSA calculation
gbsa_eps_solvent = 78.5    # solvent dielectric constant in GB
gbsa_eps_solute  = 1.0     # solute dielectric constant in GB
gbsa_salt_cons   = 0.2     # salt concentration (mol/L) in GB
gbsa_surf_tens   = 0.005   # surface tension (kcal/mol/A^2) in SA
vdw_force_switch = YES     # turn on van der Waals force switch
```

CHAPTER

SEVEN

DYNAMICS SECTION

7.1 Molecular dynamics simulations

In MD simulations, Newton's equation of motion (*F* = *ma*) is integrated numerically, where the force *F* is derived from the first derivative of the potential energy function with respect to the atomic position. To date, various integrators have been proposed.

In the velocity Verlet algorithm, the velocities are updated with

$$\mathbf{v}_i(t) = \mathbf{v}_i(t-\Delta t) + \frac{\Delta t}{m_i}\frac{\mathbf{F}_i(t)+\mathbf{F}_i(t-\Delta t)}{2},$$

and the coordinates are updated with

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t\mathbf{v}_i(t) + \frac{\Delta t^2}{m_i}\mathbf{F}_i(t).$$

These velocity/coordinate update are performed consecutively using the following procedure:

$$\begin{aligned}
\mathbf{v}_i\left(t+\frac{\Delta t}{2}\right) &= \mathbf{v}_i(t) &&+\frac{\Delta t}{2m_i}\mathbf{F}_i(t)\\
\mathbf{r}_i(t+\Delta t) &= \mathbf{r}_i(t) &&+\Delta t\mathbf{v}_i(t)\\
\mathbf{v}_i(t+\Delta t) &= \mathbf{v}_i\left(t+\frac{\Delta t}{2}\right) &&+\frac{\Delta t}{2m_i}\mathbf{F}_i(t+\Delta t)
\end{aligned}$$

In the case of multiple time step integration with r-RESPA, the force is split into slow and fast motion forces. Slow motion force is less frequently evaluated to increase the perofrmance while keeping the accuracy.

In **SPDYN**, velocity Verlet integrator and multiple time step integrator with veclotiy Verlet type are available. The users must pay attention to the **[ENSEMBLE]** section as well, because the algorithms that control the temperature and pressure are involved in the integrator. For details, see *Ensemble section*.

---

integrator *VVER / VRES*

Default : VVER

- **VVER**: velocity Verlet integrator
- **VRES**: RESPA integrator (**SPDYN** only).

timestep *Real*

Default : 0.001 (unit : ps)

Time step in the MD run. In general, timestep can be extended to 2 fs or longer, when the SHAKE, RATTLE, or SETTLE algorithms are employed. (see *Constraints section*).

nsteps *Integer*

Default : 100

Total number of steps in one MD run. If "timestep=0.001" and "nsteps=1000000" are specified, the users can carry out 1-ns MD simulation.

eneout_period *Integer*

Default : 10

Output frequency for the energy data. The trajectories are written in the log file every **eneout_period** steps during the simulation. For example, if "timestep=0.001" and "eneout_period=1000" are specified, the energy is written every 1 ps.

crdout_period *Integer*

Default : 0

Output frequency for the coordinates data. The trajectories are written in the "dcdfile" specified in the **[OUTPUT]** section every **crdout_period** steps during the simulation.

velout_period *Integer*

Default : 0

Output frequency for the velocities data. The trajectories are written in the "dcdvelfile" specified in the **[OUTPUT]** section every **velout_period** steps during the simulation.

rstout_period *Integer*

Default : 0

Output frequency for the restart file. The restart information is written in the "rstfile" specified in the **[OUTPUT]** section every **rstout_period** steps during the simulation.

stoptr_period *Integer*

Default : 10

Frequency of removing translational and rotational motions of the whole system. Note that the rotational motion is not removed when the periodic boundary condition is employed. When you use positional restraints or RMSD restraints in the simulation, you may have to take care about removal of those motions. In some cases, such restraints can generate translational or rotational momentum in the system. If the momentum is frequently removed, the dynamics can be significantly disturbed.

nbupdate_period *Integer*

Default : 10

Update frequency of the non-bonded pairlist.

elec_long_period *Integer* (**VRES** in **SPDYN** only)

Default : 1

Frequency of long-range interaction calculation.

thermostat_period *Integer* (**VRES** in **SPDYN** only)

Default : 1

Frequency of thermostat integration. It must be multiple of **elec_long_period**.

barostat_period *Integer* (**VRES** in **SPDYN** only)

Default : 1

Frequency of barostat integration. It must be multiple of **thermostat_period**.

initial_time *Real*

Default : 0.0 (unit : ps)

Initial time of the MD run. Basically, you do not need to specify a certain value. This option is useful in the case of the restart MD run, because the initial time is reset to 0 ps.

iseed *Integer*

Default : automatically generated according to the current date and time

Seed for the pseudo-random number generator. This random number seed is used in the Langevin and Bussi thermostats (see *ensemble*). If **iseed** is not specified in the control file, it is automatically generated according to the current date and time. In the restart MD run, the random number seed is taken over from rstfile. However, if **iseed** value is specified in the control file in the restart run, it is alternatively used, and the seed in rstfile is neglected.

verbose *YES / NO*

Default : NO

Turn on or off the verbose output of the log information. For example, if "verbose=YES" is specified, virial and pressure of the system are written in the log file even in the case of the NVE or NVT ensemble.

7.2 Hydrogen mass repartitioning (HMR)

In **GENESIS2.0.0beta**, we can increase the time step in NVT/NPT conditions by evaluating temperature and pressure in more accurate ways than conventional schemes. If we want to increase the time step, however, we should be careful because there could be constraint errors. To avoid the error, we recommend the user to make use of the HMR scheme with a large time step. In HMR, the mass of hydrogen atoms are increased by two to four fold whereas the bonded hevay becomes lighter to conserver the total mass.

---

hydrogen_mr *YES / NO*

Default : NO

Turn on or off the usage of HMR

hmr_ratio *Real*

Default : 3.0

Mass scaling factor of hydrogen atoms

hmr_ratio_xh1 *Real*

Default : 3.0

Mass scaling facto of hydrogen atoms in XH1 group. If it is not written, scaling factor is decided by **hmr_ratio**.

hmr_target *All / Solute*

Default : All (only when **hydrogen_mr** = *YES*)

Target of HMR application. If **hmr_target** is *Solute*, HMR is not applied to the water molecules.

7.3 Simulated annealing and heating

In simulated annelaing or heating protocol, the following keywords are additinally specified in the conventional MD simulation. In the protocol used in **GENESIS**, the target temperature is changed linearly.

---

annealing *YES / NO*

Default : NO

Turn on or off the simulated annealing or heating protocol.

anneal_period *Integer*

Default : 0

The target temperature is changed every **anneal_period** steps during the simulation.

dtemperature *Real*

Default : 0.0 (unit : Kelvin)

Magnitute of changes of the target temperature. If **dtemperature** > 0, the temperature is increased by **dtemperature** every **anneal_period** steps. If **dtemperature** < 0, the temperature is decreased.

7.4 Targeted MD and Steered MD simulations

In GENESIS, targeted MD (TMD) and steered MD (SMD) methods are available. These methods are useful to guide a protein structure towards a target. In SMD, restraint forces (or steering forces) are applied on the selected atoms, where the RMSD with respect to the target is changed during the MD simulation. The restraint force is calculated from the derivative of the RMSD restraint potential:

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

where $RMSD(t)$ is the instantaneous RMSD of the current coordinates from the target coordinates, and $RMSD_0$ is the target RMSD value. The target RMSD value is changed linearly from the initial to target RMSD values:

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

where T is the total MD simulation time. Targeted MD (TMD), originally suggested by J. Schlitter et al. *[39]*, is different from SMD in that force constants are changed during MD simulations. If the users 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, the users could find trajectories where RMSD is almost identical to the target RMSD at each time. In **[SELECTION]** section, the users select atoms involved in RMSD calculations for SMD or 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*

Default : NO

Turn on or off the targeted MD simulation.

steered_md *YES / NO*

Default : NO

Turn on or off the steered MD simulation.

initial_rmsd *Real*

Default : 0.0 (unit : Angstrom)

Initial value of the reference rmsd. If not specified explicitly, it is calculated from the initial and reference structures.

final_rmsd *Real*

Default : 0.0 (unit : Angstrom)

Final value of the reference rmsd.

Note: In the RMSD restraint, structure fitting scheme is specified in the **[FITTING]** section (see *Fitting section*). Since the default behavior was significantly changed in ver. 1.1.5 (no fitting applied on the default setting), the users of 1.1.4 or before must pay special attention on the fitting scheme. In versions of 1.1.4 or before, structure fitting is automatically applied for the atoms concerning restraint potential.

7.5 Examples

100-ps MD simulation with the velocity Verlet integrator with the timestep of 2 fs:

```
[DYNAMICS]
integrator        =   VVER  # velocity Verlet
nsteps            =  50000  # number of MD steps (100ps)
timestep          =  0.002  # timestep (2fs)
eneout_period     =    500  # energy output period (1ps)
crdout_period     =    500  # coordinates output period (1ps)
rstout_period     =  50000  # restart output period
nbupdate_period   =     10  # nonbond pair list update period
```

100-ps MD simulation with the RESPA integrator with the timestep of 2.5 fs:

```
[DYNAMICS]
integrator        =   VRES  # RESPA integrator
nsteps            =  40000  # number of MD steps (100ps)
timestep          = 0.0025  # timestep (2.5fs)
eneout_period     =    400  # energy output period (1ps)
crdout_period     =    400  # coordinates output period (1ps)
rstout_period     =  40000  # restart output period
nbupdate_period   =     10  # nonbond pair list update period
elec_long_period  =      2  # period of reciprocal space calculation
thermostat_period =     10  # period of thermostat update
barostat_period   =     10  # period of barostat update
```

The following is an example for simulated annelaing in the NVT ensemble (see *Ensemble section*), where the temperature is decreased from 500 K by 2 K every 250 steps in the 250,000-steps MD simulation (1 step = 2 fs). Thus, the temperature eventually reaches to 300 K during 50 ps. Note that heating or annealing is only available with the leap-frog integrator.

```
[DYNAMICS]
integrator       =   VVER   # leap-frog integrator
nsteps           =  25000   # number of MD steps
timestep         =  0.002   # timestep (ps)
nbupdate_period  =     10   # nonbond pair list update period
annealing        =    YES   # simulated annealing
dtemperature     =   -2.0   # delta temperature
anneal_period    =    250   # temperature change period

[ENSEMBLE]
ensemble         = NVT      # [NVT,NPT,NPAT,NPgT]
tpcontrol        = BUSSI    # [BERENDSEN,NHC,BUSSI]
temperature      = 500.0    # initial temperature (K)
```

CHAPTER

EIGHT

MINIMIZE SECTION

8.1 Energy minimization

In the **[MINIMIZE]** section, the user can select methods for energy minimization. Currently, only the steepest descent (SD) algorithm is available, Note that constraint algorithms such as SHAKE are not available in the energy minimization scheme in **GENESIS**. The energy minimization can be done with restraints (see *Restraints section*).

When the energy minimization is carried out for the initial structure, it is strongly recommended to use the option "contact_check=YES" in the **[ENERGY]** section (see *Energy section*). This is because the initial structure is usually artificial, and sometimes contains atomic clashes, where the distance between atoms is very short. Such strong interactions can generate huge forces on the atoms, resulting in unstable calculations, which might cause memory errors.

---

method *SD*

Default : **SD

Algorithm of minimization.

- **SD** : Steepest descent method

nsteps *Integer*

Default : 100

Number of minimization steps.

eneout_period *Integer*

Default : 10

Frequency of energy outputs.

crdout_period *Integer*

Default : 0

Frequency of coordinates outputs.

rstout_period *Integer*

Default : 0

Frequency of restart file updates.

nbupdate_period *Integer*

Default : 10

Frequency of non-bonded pair-list updates

8.2 Steepest descent method

force_scale_min *Real*

Default : 0.00005

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.

force_scale_max *Real*

Default : 0.0001

Maximum value of the force scaling coefficient in the steepest descent method.

8.3 Examples

A 2,000-step energy minimization with the steepest descent method:

```
[MINIMIZE]
method           = SD      # Steepest descent
nsteps           = 2000    # number of minimization steps
eneout_period    =   50    # energy output period
crdout_period    =   50    # coordinates output period
rstout_period    = 2000    # restart output period
nbupdate_period  =   10    # nonbond pair list update period
```

CHAPTER

NINE

CONSTRAINTS SECTION

9.1 SHAKE/RATTLE algorithms

In the **[CONSTRAINTS]** section, keywords related to bond constraints are specified. In the leapfrog integrator, the SHAKE algorithm is applied for covalent bonds involving hydrogen *[40]*. In the velocity Verlet and multiple time-step integrators, not only SHAKE but also RATTLE are used *[41]*. Note that bond constraint between heavy atoms is not available currently.

---

rigid_bond *YES / NO*

Default : NO

Turn on or off the SHAKE/RATTLE algorithms for covalent bonds involving hydrogen.

shake_iteration *Integer*

Default : 500

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.

shake_tolerance *Real*

Default : 1.0e-10 (unit : Angstrom)

Tolerance of SHAKE/RATTLE convergence.

hydrogen_type *NAME / MASS*

Default : NAME

This parameter defines how hydrogen atoms are detected. This parameter is ignored when *rigid_bond = NO*. Usually, the users do not need to take care about this parameter.

- **MASS** : detect hydrogen only based on the atomic mass. If the mass of an atom is less than *hydrogen_mass_upper_bound* and greater than 0, that atom is considered as a hydrogen.
- **NAME** : detect hydrogen based on the atom name, type, and mass. If the mass of an atom is less than *hydrogen_mass_upper_bound* and the name or type begins with 'h', 'H', 'd', or 'D', that atom is considered as a hydrogen.

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

o: treated as hydrogen, x: not treated as hydrogen. Here, we assumed *hydrogen_mass_upper_bound* 2.1.

hydrogen_mass_upper_bound *Real*

Default : 2.1

This parameter defines the upper limit of atomic mass to determine the hydrogen atom. For exmaple, if you define it as 3.0, the atom with the atomic mass less than 3.0 is treated as a hydrogen. You should write it in the case of hydrogen mass repartitioning scheme. This option is available in **GENESIS 1.2** or later.

9.2 SETTLE algorithm

fast_water *YES / NO*

Default : YES

Turn on or off the SETTLE algorithm for the constraints of the water molecules *[42]*. Although the default is "fast_water=YES", the users must specify "rigid_bond=YES" to use the SETTLE algorithm. If "rigid_bond=YES" and "fast_water=NO" are specified, the SHAKE/RATTLE algorithm is applied to water molecules, which is not computationally efficient.

water_model *expression or NONE*

Default : TIP3

Residue name of the water molecule to be rigidified in the SETTLE algorithm. In the case of the AMBER force field, "water_model = WAT" must be specified.

Note: TIP4P water model is availabe in **GENESIS 1.2** or later. 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.

9.3 Examples

In the case of the CHARMM force field:

```
[CONSTRAINTS]
rigid_bond  = YES   # Turn on SHAKE/RATTLE
fast_water  = YES   # Turn on SETTLE
```

In the case of the AMBER force field:

```
[CONSTRAINTS]
rigid_bond  = YES   # Turn on SHAKE/RATTLE
fast_water  = YES   # Turn on SETTLE
water_model = WAT   # residue name of the rigid water
```

Turn off all constraints in the system

```
[CONSTRAINTS]
rigid_bond  = NO
fast_water  = NO
```

CHAPTER

TEN

ENSEMBLE SECTION

10.1 Thermostat and barostat

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 Berendsen thermostat with weak coupling ("tpcontrol=berendsen"), velocities are rescaled by

$$\lambda = \left[1 + \frac{\Delta t}{\tau}\left(\frac{K_{\text{target}}}{K} - 1\right)\right]^{\frac{1}{2}}$$

at every step where τ is the dampling time *[43]*. It almost conserves dynamics of NVE and one of the most popular thermostats. However, it fails to generate canonical distributions and anomalous effects can happern *[44]*. Bussi suggested a modification of Berendsen thermostat to generate canonical distribution *[45]*. According to the scheme ("tpcontrol=bussi"), the target temperature is not a fixed value but stochastically driven from the distributions:

$$P\left(K_{\text{target}}\right) \propto {K_{\text{target}}}^{\frac{N_f}{2}-1} e^{-\beta K_{\text{target}}}.$$

In addition, random force ΔW is applied to update temperature based on the instantaneous and target temperatures:

$$\Delta K = \left(K_{\text{target}} - K\right)\frac{\Delta t}{\tau} + 2\sqrt{\frac{K K_{\text{target}}}{N_f}}\frac{\Delta W}{\tau}.$$

Another famous thermostat is suggested by Martyna et al., which is named as Nose-Hoover chain (NHC) *[46]*. This thermostat was designed to solve non-ergodic progem in Nose-Hoover thermostat. In this scheme, we solve the euqation of motion with additional textit{m} chains:

$$
\begin{aligned}
\frac{d\mathbf{r}_k}{dt} &= \frac{\mathbf{p}_k}{m_k} \\
\frac{d\mathbf{p}_k}{dt} &= \mathbf{F}_k - \frac{p_{\eta_1}}{Q_1}\mathbf{p}_k \\
\frac{d\eta_\alpha}{dt} &= \frac{p_{\eta_\alpha}}{Q_\alpha},\ \alpha = 1, 2, ..., m \\
\frac{dp_{\eta_1}}{dt} &= K - K_{\text{target}} - \frac{p_{\eta_2}}{Q_2}p_{\eta_1} \\
\frac{dp_{\eta_\alpha}}{dt} &= \frac{p_{\eta_{\alpha-1}}}{Q_{\alpha-1}} - k_B T - \frac{p_{\eta_{\alpha+1}}}{Q_{\alpha+1}}p_{\eta_\alpha},\ \alpha = 2, ..., m-1 \\
\frac{dp_{\eta_m}}{dt} &= \frac{p_{\eta_{m-1}}}{Q_{m-1}} - k_B T.
\end{aligned}
$$

In **GENESIS2.0.0beta**, we evaluate temperature in more accurate way than existing ones. With velocity Verlet integration, two kinetic energy types can be used in temperature evaluations:

$$
\begin{aligned}
K_{\text{full}} &= \sum_{k=1}^{N} \frac{p_k(t)^2}{2m_k} \\
K_{\text{half}} &= \frac{1}{2}\sum_{k=1}^{N}\left(\frac{p_k\left(t-\frac{\Delta t}{2}\right)^2}{2m_k} + \frac{p_k\left(t+\frac{\Delta t}{2}\right)^2}{2m_k}\right)
\end{aligned}
$$

These kinetic energies are accurate up to the first order of a time step and not recommended for a large time step. In **GENESIS2.0.0beta**, temperature is evaluted by combining the two kinetic energies:

$$
N_f k_B T = \frac{4}{3}\langle K_{\text{half}}(t)\rangle + \frac{2}{3}\langle K_{\text{full}}(t)\rangle
$$

Temperature evaluted in this way is accurate up to the third order of the time step.

Barostat in **GENESIS2.0.0beta** is based on MTK barostat type suggested by *[47]* with three thermostats described above. To make use of kinetic energy at $t + \frac{\Delta t}{2}$, we recommend group temperature/pressure evaluations where kinetic energy and virial are evaluted by considering XHn group one particle.

---

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

Default : NVE

Type of ensemble.

- **NVE**: Microcanonical ensemble.
- **NVT**: Canonical ensemble.
- **NPT**: Isothermal-isobaric ensemble.
- **NPAT**: Constant area A (XY), pressure along the normal (Z), temperature *[48]*. In this case, *isotropy* must be set to 'XY-FIXED' (see below).
- **NPgT**: Constant surface-tension γ (XY), pressure along the normal (Z), temperature *[48]*. In this case, *isotropy* must be set to 'SEMI-ISO' (see below).

temperature *Real*

Default : 298.15 (unit : Kelvin)

Initial and target temperature.

pressure *Real*

Default : 1.0 (unit : atm)

Target pressure in the NPT ensemble. In the case of the NPAT and NPgT ensembles, this is the pressure along the 'Z' axis.

group_tp *Yes / No*

Default : Yes

use of group temperature/pressure evaluation in thermostat/barostat

gamma *Real*

Default : 0.0 (unit : dyn/cm)

Target surface tension in NPgT ensemble.

tpcontrol *NO / BERENDSEN / NHC / BUSSI*

Default : NO

Type of thermostat and barostat. The availabe algorithm depends on the integrator.

- **NO**: Do not use temperature/pressure control algorithm (for NVE only)
- **BERENDSEN**: Berendsen thermostat with MTK barostat *[43] [47]*
- **NHC**: Nose-Hoover chain thermostat with MTK barostat *[43] [47]*
- **BUSSI**: Bussi's thermostat with MTK barostat *[45] [49] [47]*

tau_t *Real*

Default : 5.0 (unit : ps)

Temperature coupling time in the Berendsen and Bussi thermostats.

tau_p *Real*

Default : 5.0 (unit : ps)

Pressure coupling time in the Berendsen and Bussi barostats.

compressibility *Real*

Default : 0.0000463 (unit : atm^{-1})

Compressibility parameter in the Berendsen barostat.

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

Default : ISO

Isotropy of the simulation system. 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 *[50]*. 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).

10.2 Examples

NVT ensemble with Bussi thermostat:

```
[ENSEMBLE]
ensemble    = NVT        # Canonical ensemble
tpcontrol   = BUSSI      # Bussi thermostat
temperature = 300.0      # target temperature (K)
```

NPT ensemble with isotropic pressure coupling:

```
[ENSEMBLE]
ensemble    = NPT        # Isothermal-isobaric ensemble
tpcontrol   = BUSSI      # Bussi thermostat and barostat
temperature = 300.0      # target temperature (K)
pressure    = 1.0        # target pressure (atm)
```

NPT ensemble with semi-isotropic pressure coupling, which is usually used for lipid bilayer systems:

```
[ENSEMBLE]
ensemble    = NPT        # Isothermal-isobaric ensemble
tpcontrol   = BUSSI      # Bussi thermostat and barostat
temperature = 300.0      # target temperature (K)
pressure    = 1.0        # target pressure (atm)
isotropy    = SEMI-ISO   # Ratio of X to Y is kept constant
```

NPAT ensemble:

```
[ENSEMBLE]
ensemble    = NPAT       # Constant area ensemble
tpcontrol   = BUSSI      # Bussi thermostat and barostat
temperature = 300.0      # target temperature (K)
pressure    = 1.0        # target normal pressure (atm)
isotropy    = XY-FIXED   # the system area is kept constant
```

CHAPTER

ELEVEN

BOUNDARY SECTION

11.1 Boundary condition

type *PBC*

Default : PBC

Type of boundary condition.

- **PBC**: Periodic boundary condition (rectangular or cubic box)

box_size_x *Real*

Default : N/A (unit : Angstrom)

Box size along the x dimension.

box_size_y *Real*

Default : N/A (unit : Angstrom)

Box size along the y dimension.

box_size_z *Real*

Default : N/A (unit : Angstrom)

Box size along the z dimension.

Note: If the simulation system has a periodic boundary condition (**PBC**), the user must specify the box size in the control file (at the energy minimization stage in most cases). During the simulations, box size is saved in a restart file. If the restart file is used as an input of the subsequent simulation, the box size is overwritten with the restart information. Note that in this case the box size given in the control file is ignored.

11.2 Domain decomposition

domain_x *Integer*

Default : N/A (Optional) (**SPDYN** only)

Number of domains along the x dimension.

domain_y *Integer*

Default : N/A (Optional) (**SPDYN** only)

Number of domains along the y dimension.

domain_z *Integer*

Default : N/A (Optional) (**SPDYN** only)

Number of domains along the z dimension.

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 the user specifies the number of domains explicitly, please make sure that the product of the domain numbers in each dimension (i.e., domain_x * domain_y * domain_z) is equal to the total number of MPI processes.

11.3 Examples

- Simulations with the periodic boundary condition, where the box size is set to 64 x 64 x 64. In this case, the user should not use a restart file as an input, because the box size in the control is overwritten with that in the restart file.

```
[BOUNDARY]
type        = PBC        # periodic boundary condition
box_size_x = 64.0        # Box size in the x dimension (Ang)
box_size_y = 64.0        # Box size in the y dimension (Ang)
box_size_z = 64.0        # Box size in the z dimension (Ang)
```

CHAPTER

TWELVE

SELECTION SECTION

12.1 Atom selection

This section is used to select atoms, and define them as a group. The user can select atoms accorging to their name, index, residue number, segment name, and so on. The selected group index is used in other sections. For example, restraint potential can be applied on the group selected in this section, and the force constant of the potential is specified in the **[RESTRAINTS]** section. **[SELECTION]** section is also used in the GENESIS analysis tools to specify the atoms to be analyzed.

---

group*N* *expression*

The user defines selected atoms as "group1", "group2", ..., and "group:math:*N*". Here, *N* must be a positive integer ($N \geq 1$). The user selects atoms by using keywords and operators with a certain syntax (see table below). Note that in the table mname (or moleculename, molname) is a molecule name defined by **mol_name**.

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

The user defines a molecule by specifying its segment name, first and last residue numbers, and residue name. `N` must be a positive integer ($N \geq 1$). The syntax for the residue selection is as follows:

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

For details, see the example below.

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

Table. Available keywords and operators in `group` (continued).

expression	meaning	example	other available expression
X around: *r*	atoms around *r* Angstrom of *X*	see below	around_atoms
X around_res: *r*	residues around *r* Angstrom of *X*	see below	around_residues
X around_mol: *r*	molecules around *r* Angstrom of *X*	see below	around_molecules

Note: `ai` and `atno` are slightly different. `ai` indicates the atom index which is sequentially re-numbered over all atoms in the system. On the other hand, `atno` is the index of atoms in the PDB file. Atom index in PDB file (column 2) does not always start from 1, nor is numbered sequentially. In such cases, `atno` is useful to select atoms, although it is a very rare case.

Note: Atoms that are within a distance of a given atom (X) can be selected by `around`. Note that the coordinates in `reffile` is used to judge the distance. If `reffile` is not present, those in input files (`pdbfile`, `crdfile`, etc.) are used instead. Coordinates in `rstfile` are never used.

12.2 Examples

Select atoms based on their atom name, residue name, or residue number:

```
[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)
```

Select atoms around an atom X. In the following examples, X = atom number 100.

```
[SELECTION]
group1      = atno:100 around:10.0
group2      = atno:100 around_res:10.0
group3      = atno:100 around_mol:10.0
group4      = atno:100 around_mol:10.0 or atno:100
```

In group1, atoms around 10.0 Angstrom of X are selected. Group 2 selects residues around 10.0 Angstrom of X, i.e., if the distance between X and any one of atoms in a residue is less than 10.0 Angstrom, all atoms of the residue are selected. Group 3 is the same as group 2, but for a molecule. Note that these commands do NOT select X itself. In order to include X in the selection, add "or atno:100", as in group 4.

Select atoms around multiple atoms.

```
[SELECTION]
group1      = atno:100-101 around:10.0
group2      = (sid:PROT around_res:10.0) and rnam:TIP3
group3      = (rno:1 around:10.0) or rno:1
```

Group 1 selects atoms around 10.0 Angstrom of atom 100 *or* 101. Note that it is NOT "100 *and* 101" nor a center of 100 and 101. Group 2 is an example to select water molecules around a protein (segname PROT). Group 3 selects not only the atoms around residue1 but also the atoms of residue1.

CHAPTER

THIRTEEN

RESTRAINTS SECTION

13.1 Restraint potential

[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.

13.1.1 General keywords

nfunctions *Integer*

Default: 0

Number of restraint functions.

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

Default: N/A

Type of restraint.

- **POSI**: positional restraint. The reference coordinates are given by **reffile**, **ambreffile**, or **groreffile** in **[INPUT]**. (see *Input section*)
- **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 section*) 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. This option requires *modefile* in the *Input section*.

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*

Default: 0.0 (unit: depend on the restraint type)

Force constant of a restraint function. The unit depends on the type of restraint. Namely, $\mathrm{kcal/mol/\AA}^n$ is used in the case of *DIST* and *RMSD*, while $\mathrm{kcal/mol/rad}^n$ in the case of *ANGLE* and *DIHED*, where n is **exponent*N*** specified in this section.

reference*N* *Real*

Default: 0.0 (unit: depend on the restraint type)

Reference value of a restraint function. For the positional restraint, the value is ignored. The unit depends the type of restraint. Namely, Angstrom is used in the case of *DIST*, while degree (NOT radian) is used in the case of *ANGLE* and *DIHED*.

select_index*N* *Integer*

Default: N/A

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

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 the 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*

Default : ALL

Direction of the *POSI* restraint. If *X* or *Y* or *Z* is specified, restraints along the other two axes are not applied.

exponent*N* *Integer*

Default : 2

Exponent factor of the restraint function. The default is the harmonic. 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)

Default : 1

Exponent factor used in the distance sum calculations. 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_dist*N***, and w is **weight_dist*N***.

weight_dist*N* *Real* (**DIST[MASS]** only)

Default : 1.0

Weight factor used in the distance sum calculations.

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`.

13.1.2 Pressure derived from restraints

Basically, the pressure calculated from the restraint potential is included in an internal pressure, which is kept constant during the simulation in the NPT ensemble. However, the pressure derived from positional and RMSD restraints are treated as an external pressure by default. Keywords **pressure_position** and

pressure_rmsd are used to include those pressures in the internal pressure. If the simulation with POSI or RMSD restraint showed a strange behaviour (especially, when a strong force constant is applied), turn on these options.

pressure_position *YES / NO*

Default : NO

The virial terms from positional restraints are included in pressure evaluation.

pressure_rmsd *YES / NO*

Default : NO

The virial terms from RMSD restraints are included in pressure evaluation.

13.1.3 Advanced definition of restraints

Restraints can be also defined in an external input file (**localresfile**). In this case, number of local restraints must NOT be included in **nfunctions**. This option is availabe in **SPDYN** only. For details, see *Input section*.

13.1.4 Restraints in REUS simulations

If you employed a certain restraint term for REUS runs, *nreplica* of force constants and reference values must be given as a space-separated list. The above keywords except for **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. For details, see *REMD section*.

13.2 Examples

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

FOURTEEN

FITTING SECTION

14.1 Structure fitting

(In GENESIS 1.1.5 or later only) Keywords in **[FITTING]** section define a structure superimposition scheme, which is often employed in targeted MD, steered MD, or String method (see *RPATH section*) with positional restraint. In the String method, the reference coordinate for fitting is given by **fitfile** in the **[INPUT]** section. Otherwise (MD, MIN, REMD), the reference coordinate is given by **reffile**, **ambreffile**, or **groreffile** in the **[INPUT]** section. Note that this section is not related to the cryo-EM flexible fitting (see Experiments)

---

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

Default: TR+ROT

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

fitting_atom: *Integer*

Default: N/A

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 section*). For example, if you specify `fitting_atom = 1`, the reference atoms are members of `group1` in the **[SELECTION]**.

mass_weight: *YES / NO*

Default: 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 the String method, mass-weighted superimposition is not supported.

force_no_fitting: *YES / NO*

Default: NO

This parameter must not be changed for standard MD runs. If the parameter is set to YES and fitting_method is set to NO, the fitting routine is turned off. 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 simulations involving RMSD calculation (targeted/steered MD, for example). But such fitting is not desirable when generating initial structure set for the String method using Cartesian coordinate as CV (see *RPATH section*). Actually, *fitting_method = NO* was implemented just for this specific purpose. If you are really want to turn off fittings of RMSD calculation for preparation of initial structure set for String method, please specify *fitting_method = NO* and *force_no_fitting = YES*.

14.2 Examples

Example of **[FITTING]** section

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

CHAPTER

FIFTEEN

REMD SECTION

15.1 Replica-exchange molecular-dynamics simulation (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 *[51] [52]*. 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:

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

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 *[53] [54]*. 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 *[55]*, surface-tension REMD *[56]*, REUS (or Hamiltonian REMD) *[57] [58]*, replica exchange with solute tempering (REST) *[59] [60]*, and their multi-dimensional combinations are available in both **ATDYN** and **SPDYN**. Basically, these methods can be employed in the NVT, NPT, NPAT, NPgT ensembles, except for the surface-tension REMD, which is only used in the NPgT ensemble. REMD simulations in GENESIS require an MPI environment. At least one MPI process must be assigned to one replica. For example, when the user wants to employ 32 replicas, $32n$ MPI processes are required.

In the following parameters excluding *dimension*, *exchange_period*, and *iseed*, the last character 'N' must be replaced with 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. For details, see the examples below.

---

dimension *Integer*

Default: 1

Number of dimensions (i.e. number of parameter types to be exchanged)

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

Default: TEMPERATURE

Type of parameter to be exchanged in the N-th dimension

- **TEMPERATURE**: Temperature REMD *[51]*
- **PRESSURE**: Pressure REMD *[55]*
- **GAMMA**: Surface-tension REMD *[56]*
- **RESTRAINT**: REUS (or Hamiltonian REMD) *[57] [58]*
- **REST**: replica exchange with solute tempering (REST2 or gREST) *[59] [60]*, which is totally different from the original version of REST *[61]*. Currently, only AMBER and CHARMM force fields are supported.

nreplica*N* *Integer*

Default: 0

Number of replicas (or parameters) in the N-th dimension

For fast REMD calculation on Fugaku, we can write **nreplicax*N***, **nreplicay*N***, and **nreplicaz*N*** such that their multiplication is same as **nreplica*N***.

parameters*N* *Real*

Default: N/A

List of parameters for each replica in the N-th dimension. Parameters must be given as a space-separated list, and the total number of parameters must be equal to **nreplicaN**. In case of REUS (type = RESTRAINT), parameters must be specified in **[RESTRAINTS]** section (see the sample below). In case of gREST (type = REST), these parameters are considered as temperature of solute region. Note that the order of the parameters in this list must NOT be changed before and after the restart run, even if the parameters are exchanged during the REMD simulation.

exchange_period *Integer*

Default: 100

Frequency of the parameter exchange attempt. If "exchange_period = 0" is specified, REMD simulation is carried out without parameter exchange, which is useful to equilibrate the system in a condition assinged to each replica before performing the production run.

cyclic_params*N* *YES / NO*

Default: NO

Turn on or off the periodicity of the parameters in the N-th dimension. If "cyclic_paramsN = YES" is specified, the first and last parameters are considered as neighbouring parameters. This option can be applicabe to all parameter types. Basically, this is useful in the case of REUS in dihedral angle space, since the dihedral angle has a periodicity.

iseed *Integer*

Default: 3141592

Random number seed in the replica exchange scheme. If this is not specified explicitly, iseed is taken over from the restart file.

Note: In the [ENSEMBLE] section, there is also a parameter "temperature". In the T-REMD simulation, this temperature is ignored, even if it is specified explicitly. Similarly, pressure and gamma in the [ENSEMBLE] section are ignored in the P-REMD and surface-tension REMD simulations, respectively.

Note: When multi-dimensional REMD is carried out, parameters are exchanged alternatively. For example, in TP-REMD (type1 = TEMPERATURE and type2 = PRESSURE), there is a temperature exchange first, followed by a pressure exchange. This is repeated during the simulations.

15.2 Replica-exchange umbrella-sampling (REUS)

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

Index of the restraint function to be used in the REUS simulation. The detailed parameters in the restraint function (e.g., force constant and reference) are defined in the **[RESTRAINTS]** section (see *Restraints section*). Note that the order of the parameters in the **[RESTRAINTS]** section must NOT be changed before and after the restart run, even if the parameters are exchanged during the REUS simulation.

GENESIS supports not only on-grid but also off-grid schemes. In the off-grid REUS, multiple restraints are merged into a single reaction coordinate (see example below). Those restraints are defined in **[RESTRAINTS]** section, where the number of parameters (const and reference) must be equal to *nreplicaN*. 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 currently.

Note: Positional restraint is not available for REUS. In **SPDYN**, PCA restraint is not available for REUS. The control file format was completely changed after verion 1.1.0, since the off-grid REUS

scheme was introduced. When the users use the old control file, please be careful.

15.3 Replica-exchange with solute-tempering (gREST)

select_index*N* *Integer*

Default: N/A

Index of an atom group. The selected atoms are considered as "solute" in gREST. The index must be defined in **[SELECTION]** (see *Selection section*).

analysis_grest *YES / NO*

Default: NO

A logical flag to do energy analysis during MD simulations.

param_type*N* *ALL / BOND / ANGLE / UREY / DIHEDRAL / IMPROPER / CMAP / CHARGE / LJ*

Default: ALL

Solute energy terms for gREST *[60]* 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 examples). The parameter names are case-insensitive as follows:

- **ALL**: all the available energy terms.
- **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.
- **LJ**: (aliases: **L**, **LJS**): Lennard-Jones interaction terms.

Note: Note that restraint energy terms defined in **[RESTRAINTS]** cannot be treated as solute terms. They never be affected by gREST solute temperatures. In SPDYN, water atoms cannot be specified as "solute" now. This limitation will be removed in the future version.

Note: When the coulombic interaction terms are considered as the solute, the solute region should have a net charge of 0 for an adequate PME calculation.

15.4 Examples

Basically, REMD simulations in **GENESIS** can be carried out by just adding the **[REMD]** section in the control file of a normal MD simulation. For details, see the online Tutorial (https://www.r-ccs.riken.jp/labs/cbrt/tutorials2019/).

15.4.1 T-REMD

If the users want to carry out T-REMD simulations with 4 replicas in the NVT ensemble, where each replica has the temperature 298.15, 311.79, 321.18, or 330.82 K, and replica exchange is attempted every 1000 steps, the following section is added to the control file of a normal MD simulation in the NVT ensemble:

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

As for the T-REMD simulation in the NPT ensemble, the users add this section to the control file of a normal MD simulation in the NPT ensemble. The REMD temperature generator (http://folding.bmc.uu.se/remd/) is a useful tool to set the target temperature of each replica.

15.4.2 Two-dimensional REMD (T-REMD/REUS)

The following is an example of two-dimensinal REMD, where temperature and restraint are exchanged alternatively, The 1st dimension is T-REMD with 8 parameters, and 2nd dimension is REUS in distance space with 4 parameters. In total, 8 x 4 = 32 replicas are used:

```
[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:25
group2          = ai:392

[RESTRAINTS]
nfunctions      = 1
function1       = DIST
constant1       =  2.0   2.0   2.0   2.0
reference1      = 10.0  10.5  11.0  11.5
select_index1   = 1 2
```

These sections are added to the control file of a normal MD simulation.

15.4.3 Off-grid REUS

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

[SELECTION]
group1          = ai:25
group2          = ai:392
group3          = ai:72
group4          = ai:73
group5          = ai:74
group6          = ai:75

[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
```

15.4.4 gREST

In this example, the dihedral, CMAP, and LJ energy terms in the selected atom groups are treated 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

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

T-REMD in the two-dimensional REMD (T-REMD/REUS) may be replaced with gREST (gREST/REUS *[62]*) to reduce the required number of replicas.

```
[REMD]
dimension       = 2
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   = 3
type2           = RESTRAINT
nreplica2       = 4
rest_function2  = 1

[SELECTION]
group1          = ai:25
group2          = ai:392
group3          = ai:1-313

[RESTRAINTS]
nfunctions      = 1
function1       = DIST
constant1       =  2.0   2.0   2.0   2.0
reference1      = 10.0  10.5  11.0  11.5
select_index1   = 1 2
```

CHAPTER

SIXTEEN

RPATH SECTION

16.1 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 *[63]*, the on-the-fly string method *[64]*, and the string method of swarms of trajectories *[65]*. Among these algorithgms, the mean forces string method is available in **SPDYN** *[66]*.

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*

Default: 1

Number of replicas (images) for representing the pathway.

rpath_period *Integer*

Default: 0

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.

delta *Real*

Default: 0.0

Step-size for steepest descent update of images.

smooth *Real*

Default: 0.0

Smoothing parameter which controls the aggressiveness of the smoothing. Values from 0.0 to 0.1 are recommended, where "smooth = 0.0" means no-smoothing

rest_function *List of Integers*

Default: N/A

List of restraint function indices defined in **[RESTRAINTS]** section (see *Restraints section*). 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*

Default: 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.

use_restart *YES / NO*

Default : YES

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.

Note: 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.

16.2 Examples

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
```

CHAPTER

SEVENTEEN

GAMD SECTION

17.1 Gaussian accelerated Molecular Dynamics

In the **[GAMD]** section, the users can specify keywords for Gaussian accelerated Molecular Dynamics (GaMD) simulation. The GaMD method *[67][68]* accelerates the conformational sampling of biomolecules by adding a harmonic boost potential to smooth their potential energy surface. GaMD has the advantage that reaction coordinates do not need to be predefined, thus setting up the system for the simulation is rather easy. The use of the harmonic boost potential allows to recover the unbiased free-energy changes through cumulant expansion to the second order, which resolves the practical reweighting problem in the original accelerated MD method.

GaMD was developed as a potential-biasing method for enhanced sampling. It accelerates the conformational sampling of a biomolecule by adding a non-negative boost potential to the system potential energy $U(\vec{x})$:

$$U'(\vec{x}) = U(\vec{x}) + \Delta U^{\mathrm{GaMD}}\left(U(\vec{x})\right),$$

where $\vec{x}$ is the configuration of the system, $U'(\vec{x})$ is the modified potential energy, and ΔU^{GaMD} is the boost potential depending only on $U(\vec{x})$.

In conventional accelerated MD *[69][70][71]*, the average of the Boltzmann factors of the boost potential terms appears in the reweighting equation of the probability along the selected reaction coordinates, causing a large statistical error. In order to reduce the noise, GaMD uses a harmonic boost potential, which adopts a positive value only when the system potential is lower than an energy threshold E:

$$\Delta U^{\mathrm{GaMD}}\left(U(\vec{x})\right) = \begin{cases} \frac{1}{2}k\{E - U(\vec{x})\}^2 & (U(\vec{x}) < E) \\ 0 & (U(\vec{x}) \geq E) \end{cases},$$

where k is a harmonic force constant. $U'(\vec{x})$ should satisfy the following relationships *[67][68]*: $U'(\vec{x}_1) < U'(\vec{x}_2)$ and $U'(\vec{x}_2) - U'(\vec{x}_1) < U(\vec{x}_2) - U(\vec{x}_1)$ if $U(\vec{x}_1) < U(\vec{x}_2)$. To keep the relationships, the threshold energy needs to be set as:

$$U_{\max} \leq E \leq U_{\min} + \frac{1}{k},$$

where $U_{\max}$ and $U_{\min}$ are maximum and minimum energies of the system, respectively. To ensure accurate reweighting, the deviation of the potential must also satisfy the relation:

$$k(E - U_{\mathrm{ave}})\sigma_U \leq \sigma_0,$$

where U_{ave} and σ_U are the average and standard deviation of $U(\vec{x})$, respectively. σ_0 is a user-specified upper limit. k_0 is defined as $k_0 \equiv k(U_{\max} - U_{\min})$, then $0 < k_0 \leq 1$.

When E is set to the lower bound $U_{\max}$, k_0 is determined by

$$k_0 = \min\left(1, \frac{\sigma_0}{\sigma_U}\frac{U_{\max} - U_{\min}}{U_{\max} - U_{\mathrm{ave}}}\right)$$

When E is set to the upper bound $U_{\min} + 1/k$, k_0 is set to

$$k_0'' \equiv \left(1 - \frac{\sigma_0}{\sigma_U}\right)\frac{U_{\max} - U_{\min}}{U_{\mathrm{ave}} - U_{\min}}$$

if $0 < k_0'' < 1$, and k_0 is set to 1 otherwise.

The above parameters ($U_{\max}$, $U_{\min}$, U_{ave}, and σ_U) are determined from short-time simulations a priori. When the distribution of the boost potential approaches Gaussian distribution, the cumulant expansion of the average of $\exp[\beta\Delta U^{\mathrm{GaMD}}]$ to the second order provides a good approximation for the free energy *[72]*.

GaMD can be combined with REUS in such a way that each replica in REUS is accelerated by the GaMD boost potential:

$$\begin{aligned} U_i''(\vec{x}) &= U'(\vec{x}) + \Delta U_i^{\mathrm{REUS}}\left(\xi(\vec{x})\right) \\ &= U(\vec{x}) + \Delta U^{\mathrm{GaMD}}\left(U(\vec{x})\right) + \Delta U_i^{\mathrm{REUS}}\left(\xi(\vec{x})\right), \end{aligned}$$

where $U_i''(\vec{x})$ is the modified potential energy of replica i, $\Delta U_i^{\mathrm{REUS}}$ is the bias potential of REUS for replica i, and $\xi(\vec{x})$ is the collective variable of REUS. This method is referred to as Gaussian accelerated replica exchange umbrella sampling (GaREUS) *[73]*. The parameters in the GaMD boost potential are used in all replicas of GaREUS simulations. By using this combination, the simulated system in each replica becomes more flexible, or the energy barrier irrelevant to the collective variable is lowered, enhancing the sampling efficiency. When performing GaREUS simulations, the user must specify [REMD] section to use REUS and define a collective variable in the [SELECTION] and [RESTRAINTS] sections. Please check the example below.

---

gamd *YES / NO*

Default : NO

Enable the GaMD method.

boost *YES / NO*

Default : YES

Flag to apply GaMD boost to the system (). If *boost = NO*, boost is not applied but GaMD parameters are updated from the trajectory.

boost_type *DUAL / DIHEDRAL / POTENTIAL*

Default: DUAL

Type of boost.

- **DUAL**: Boost is applied on both the dihedral and total potential energies.

- **DIHEDRAL**: Boost is applied on only the dihedral energy.
- **POTENTIAL**: Boost is applied on only the total potential energy.

thresh_type *LOWER / HIGHER*

Default: LOWER

Type of threshold.

- **LOWER**: E is set to the lower bound $E = U_{\text{max}}$.
- **HIGHER**: E is set to its upper bound $E = U_{\text{min}} + 1/k$.

update_period *Integer*

Default: 0

Period of updating parameters in units of time step.

sigma0_pot *Real*

Default: 6.0 (unit: kcal/mol)

Upper limit of the standard deviation of the total potential boost (σ_0^{pot}) that allows for accurate reweighting.

pot_max *Real*

Default: -99999999.0 (unit: kcal/mol)

Maximum of the total potential energy of the system $U_{\text{max}}^{\text{pot}}$.

pot_min *Real*

Default: 99999999.0 (unit: kcal/mol)

Minimum of the total potential energy of the system $U_{\text{min}}^{\text{pot}}$.

pot_ave *Real*

Default: 0.0 (unit: kcal/mol)

Average of the total potential energy of the system $U_{\text{ave}}^{\text{pot}}$.

pot_dev *Real*

Default: 0.0 (unit: kcal/mol)

Standard deviation of the total potential energy of the system σ_U^{pot}.

sigma0_dih *Real*

Default: 6.0 (unit: kcal/mol)

Upper limit of the standard deviation of the dihedral boost (σ_0^{dih}) that allows for accurate reweighting.

dih_max *Real*

Default: -99999999.0 (unit: kcal/mol)

Maximum of the dihedral energy of the system $U_{\text{max}}^{\text{dih}}$.

dih_min *Real*

Default: 99999999.0 (unit: kcal/mol)

Minimum of the dihedral energy of the system $U_{\mathrm{min}}^{\mathrm{dih}}$.

dih_ave *Real*

Default: 0.0 (unit: kcal/mol)

Average of the dihedral energy of the system $U_{\mathrm{ave}}^{\mathrm{dih}}$.

dih_dev *Real*

Default: 0.0 (unit: kcal/mol)

Standard deviation of the dihedral energy of the system σ_U^{dih}.

17.2 Examples

Example of a GaMD simulation to determine initial parameters. To obtain the initial guess of the boost potential, (pot_max, pot_min, pot_ave, pot_dev, dih_max, dih_min, dih_ave, dih_dev) are calculated from a short simulation without boosting.

```
[GAMD]
gamd          = yes
boost         = no
boost_type    = DUAL
thresh_type   = LOWER
sigma0_pot    = 6.0
sigma0_dih    = 6.0
update_period = 50000
```

Example of a GaMD simulation updating parameters. The boost potential is updated every *update_period* during the simulation.

```
[GAMD]
gamd          = yes
boost         = yes
boost_type    = DUAL
thresh_type   = LOWER
sigma0_pot    = 6.0
sigma0_dih    = 6.0
update_period = 500
pot_max       = -20935.8104
pot_min       = -21452.3778
pot_ave       = -21183.9911
pot_dev       = 78.1207
dih_max       = 16.4039
dih_min       = 8.5882
dih_ave       = 11.0343
dih_dev       = 1.0699
```

Example of a GaMD simulation for production. In order to fix the parameters (pot_max, pot_min, pot_ave, pot_dev, dih_max, dih_min, dih_ave, dih_dev), *update_period* is set to 0.

```
[GAMD]
gamd          = yes
```

```
boost          = yes
boost_type     = DUAL
thresh_type    = LOWER
sigma0_pot     = 6.0
sigma0_dih     = 6.0
update_period = 0
pot_max        = -20669.2404
pot_min        = -21452.3778
pot_ave        = -20861.5224
pot_dev        = 48.9241
dih_max        = 23.2783
dih_min        = 8.5882
dih_ave        = 13.3806
dih_dev        = 1.7287
```

Example of a GaREUS simulation. The same GaMD parameters are applied in each replica of REUS. After the simulation, the two-step reweighting procedure using the multistate Bennett acceptance ratio method and the cumulant expansion for the exponential average is required to obtain the unbiased free-energy landscapes.

```
[REMD]
dimension         = 1
exchange_period   = 5000
type1             = RESTRAINT
nreplica1         = 4
rest_function1    = 1

[GAMD]
gamd           = yes
boost          = yes
boost_type     = DUAL
thresh_type    = LOWER
sigma0_pot     = 6.0
sigma0_dih     = 6.0
update_period = 0
pot_max        = -26491.7344
pot_min        = -27447.4316
pot_ave        = -26744.5742
pot_dev        = 52.5674
dih_max        = 135.8921
dih_min        = 91.2309
dih_ave        = 116.8572
dih_dev        = 3.6465

[SELECTION]
group1 = rno:1  and an:CA
group2 = rno:10 and an:CA

[RESTRAINTS]
nfunctions     = 1
function1      = DISTMASS
constant1      = 1.0 1.0 1.0 1.0
reference1     = 5.0 6.0 7.0 8.0
select_index1 = 1 2
```

CHAPTER

EIGHTEEN

TROUBLE SHOOTING

The followings are representative error messages that the users can frequently encounter during the simulations. We describe possible reasons for each error message, and provide suggestions to solve the problem.

Compute_Shake> SHAKE algorithm failed to converge

This message indicates that constraint for the rigid bond using the SHAKE algorithm (see *Constraints section*) was failed due to some reasons. In most cases, SHAKE errors are originated from insufficient equilibration, bad initial structure, or bad input parameters. We recommend the users to check the following points:

- Reconsider the equilibration scheme. More moderate equilibration might be needed. For example, heating the system from 0 K, using a shorter timestep (e.g., 1.0 fs), or performing long energy minimization is a possible solution.
- Check the initial structure very carefully. One of the frequent mistakes in the initial structure modeling is "ring penetration" of covalent bonds. One covalent bond might be somehow inserted into an aromatic ring. Solve the ring penetration first, and then try the simulation again.
- Some force field parameters are missing or wrong, which can easily cause unstable simulations.

Check_Atom_Coord> Some atoms have large clashes

This message indicates that there is an atom pair whose distance is zero or close to zero. Those atom indexes and distance are displayed in a warning message: "WARNING: too short distance:". This situation is not allowed, especially in **SPDYN**, since it can cause a numerical error in the lookup table method. Check the initial structure first. Even if you cannot see such atomic clashes, there may be a clash between the atoms in the unit cell and image cells in the case of the periodic boundary condition. One of the automatic solutions is to specify "contact_check = YES" in the control file (see *Energy section*). However, this cannot work well, if the distance is exactly zero. In such cases, the problem should be solved by the users themselves. For example, the users may have to slightly move the clashing atoms manually, or specify larger or smaller box size, or rebuild the initial structure more carefully.

Setup_Processor_Number> Cannot define domains and cells. Smaller MPI processors, or shorter pairlistdist, or larger boxsize should be used

This message indicates that the total number of MPI processors used in your calculation is not appropriate for your system. The users had better to understand relations between the system size and number of MPI processors. In **SPDYN**, the system is divided into several domains for parallel computation, where the number of domains must be equal to

the number of MPI processors (see *Available Programs*). In most cases, this message tells you that the system could not be divided into the specified number of domains. Although there are mainly three solutions for this problem, first one is the most recommended way:

- Use smaller number of MPI processors. If it can work, the previous number was too large to handle the system.
- Use shorter pairlistdist. This treatment can make a domain size smaller, allowing to use a larger number of MPI processors. However, this is not recommended, if you are already using a recommended parameter set for switchdist, cutoffdist, and pairlistdist (e.g., 10, 12, and 13.5 Å in the CHARMM force field)
- Build a larger initial structure by adding solvent molecules in the system, which may allow the users to divide the system into the desired number of domains.

Update_Boundary_Pbc> too small boxsize/pairdist. larger boxsize or shorter pairdist should be used.

This message indicates that your system is too small to handle in the periodic boundary condition. In **ATDYN**, cell-linked list method is used to make non-bonded pairlists, where the cell size is determined to be close to and larger than the pairlist distance given in the control file. In addition, the total number of cells in x, y, and z dimensions must be at least three. **SPDYN** has a similar lower limitation in the available box size. Therefore, in order to solve this problem, the users may have to set a shorter pairlistdist, or build a larger system by adding much solvent molecules.

Compute_Energy_Experimental_Restraint_Emfit> Gaussian kernel is extending outside the map box

This message indicates that the simulated densities were generated outside the target density map. If atoms to be fitted are located near the edge of the target density map, this error can frequently happen.

- Create a larger density map by adding an enough margin to the map, which can be easily accomplished with the "voledit" tool in SITUS (https://situs.biomachina.org/).
- Examine a normal MD simulation by turning off the EM biasing potential (emfit = NO). If the simulation is not stable, there is an issue in the molecular mechanics calculation rather than the biasing potential calculation. In such cases, please check the initial structure carefully. There might be large clashes between some atoms, which can cause explosion of the target molecule, and push some atoms out of the density map. The problems to be solved are almost same with those in the SHAKE errors (see above).

Compute_Energy_Restraints_Pos> Positional restraint energy is too big

This message indicates that some atoms to be restrained are significantly deviated from the reference position, indicating that the restraint might not be properly applied to such atoms. This situation is not allowed in **SPDYN**.

- Use a larger force constant to keep their position near the reference.
- Turn off the positional restraint for such atoms if it is not essential.

CHAPTER

NINETEEN

APPENDIX

19.1 Install the requirements for Linux

In the first sub-section, we explain how to install the requirements using the package manager "apt" in Ubuntu/Debian. If you want to install them from the source codes, or if you want to use other Linux systems like CentOS, please see the second sub-section.

19.1.1 For Ubuntu/Debian users

GNU compilers and build tools

First, we install compilers and build tools.

```
$ sudo apt update
$ sudo apt install build-essential
$ sudo apt install gfortran autoconf automake
```

OpenMPI

Then, we install OpenMPI. Note that the development version (XXX-dev) should be installed.

```
$ sudo apt install openmpi-bin libopenmpi-dev

$ which mpirun mpif90 mpicc
/usr/bin/mpirun
/usr/bin/mpif90
/usr/bin/mpicc
```

LAPACK/BLAS libraries

Finally, we install LAPACK/BLAS libraries. Again, development version (XXX-dev) is installed.

```
$ sudo apt install liblapack-dev

$ ls /usr/lib/x86_64-linux-gnu/liblapack.*
$ ls /usr/lib/x86_64-linux-gnu/libblas.*
```

19.1.2 For CentOS/RedHat users

Here, we explain how to install OpenMPI and LAPACK/BLAS libraries from the source codes. We assume that the users already installed GNU compilers. The following schemes are commonly applicable to typical Linux systems including CentOS and Red Hat.

OpenMPI

The source code of OpenMPI is availabe in https://www.open-mpi.org/. The following commands install OpenMPI 3.1.5 in the user's local directory "$HOME/Software/mpi" as an example. Here, we use GNU compilers (gcc, g++, and gfortran).

```
$ cd $HOME
$ mkdir Software
$ cd Software

$ mkdir build
$ cd build

$ wget https://download.open-mpi.org/release/open-mpi/v3.1/openmpi-3.1.5.
↪tar.gz

$ tar -xvf openmpi-3.1.5.tar.gz
$ cd openmpi-3.1.5

$ ./configure --prefix=$HOME/Software/mpi CC=gcc CXX=g++ F77=gfortran␣
↪FC=gfortran

$ make all
$ make install
```

The following information is added in "~/.bash_profile" (or "~/.bashrc").

```
MPIROOT=$HOME/Software/mpi
export PATH=$MPIROOT/bin:$PATH
export LD_LIBRARY_PATH=$MPIROOT/lib:$LD_LIBRARY_PATH
export MANPATH=$MPIROOT/share/man:$MANPATH
```

Launch another terminal window or reload "~/.bash_profile" (or "~/.bashrc"):

```
$ source ~/.bash_profile
```

The OpenMPI tools should be installed in "$HOME/Software/mpi/bin".

```
$ which mpirun mpif90 mpicc
~/Software/mpi/bin/mpirun
~/Software/mpi/bin/mpif90
~/Software/mpi/bin/mpicc
```

If you want to uninstall OpenMPI, just remove the directory "mpi" in "Software".

LAPACK/BLAS libraries

The source code of LAPACK/BLAS is availabe in http://www.netlib.org/lapack/. The following commands install LAPACK 3.8.0 in the user's local directory "$HOME/Software/lapack-3.8.0" as an example. The BLAS library is also installed. We use GNU compilers (gcc and gfortran).

```
$ cd $HOME/Software
$ wget http://www.netlib.org/lapack/lapack-3.8.0.tar.gz
$ tar -xvf lapack-3.8.0.tar.gz
$ cd lapack-3.8.0

$ cp make.inc.example make.inc
$ make blaslib
$ make lapacklib

$ ls lib*
liblapack.a  librefblas.a

$ ln -s librefblas.a ./libblas.a
```

The following information is added in "~/.bash_profile" (or "~/.bashrc").

```
export LAPACK_PATH=$HOME/Software/lapack-3.8.0
```

Launch another terminal window or reload "~/.bash_profile" (or "~/.bashrc"):

```
$ source ~/.bash_profile
```

If you want to uninstall LAPACK/BLAS, just remove the directory "lapack-3.8.0" in "Software".

19.2 Install the requirements for Mac

We recommend the Mac users to utilize "Xcode" for the installation of GENESIS, and also to install "OpenMPI" from the source code to avoid a "clang" problem (see below).

19.2.1 Install general tools

Xcode and Homebrew

"Xcode" is available in the Mac App Store (https://developer.apple.com/xcode/), and it is free of charge. After the installation of Xcode, all tasks described below will be done on "Terminal". The "Terminal app" is in the "Utilities" folder in Applications. Please launch the Terminal. This terminal is almost same with that in Linux.

We recommend you to further install "Homebrew", which enables easy installation of various tools such as compilers. If you have already installed "MacPorts", you do not need to install "Homebrew" to avoid a conflict between "Homebrew" and "MacPorts". In the Homebrew website (https://brew.sh/), you can find a long command like "`/usr/bin/ruby -e "$(curl -fsSL https://...`". To install homebrew, execute that command in the Terminal prompt.

GNU compilers and build tools

First, we install "gcc", "autoconf", "automake", and other tools via homebrew:

```
$ brew install gcc
$ brew install autoconf
$ brew install automake
$ brew install wget
```

To confirm the installation of "gcc", let us type the following commands:

```
$ which gcc
/usr/bin/gcc

$ gcc --version
...
Apple LLVM version 10.0.1 (clang-1001.0.46.4)
```

These messages tell us that "gcc" is installed in the "/usr/bin" directory. However, this gcc is not a "real" GNU compiler, and it is linked to another compiler "clang". If you use this gcc for the installation of OpenMPI, it can cause a trouble in compiling **GENESIS** with a certain option. Therefore, you have to use a "real" GNU compiler, which is actually installed in "/usr/local/bin". For example, if you have installed gcc ver. 9, you can find it as "gcc-9" in "/usr/local/bin".

```
$ ls /usr/local/bin/gcc*
/usr/local/bin/gcc-9      /usr/local/bin/gcc-ar-9      ...

$ gcc-9 --version
gcc-9 (Homebrew GCC 9.2.0) 9.2.0
```

19.2.2 Install libraries

OpenMPI

We then install "OpenMPI". We specify "real" GNU compilers explicitly in the configure command. The following commands install OpenMPI in the user's local directory "$HOME/Software/mpi".

```
$ cd $HOME
$ mkdir Software
$ cd Software
$ mkdir build
$ cd build

$ wget https://download.open-mpi.org/release/open-mpi/v3.1/openmpi-3.1.5.
↪tar.gz

$ tar -xvf openmpi-3.1.5.tar.gz
$ cd openmpi-3.1.5

$ ./configure --prefix=$HOME/Software/mpi CC=gcc-9 CXX=g++-9 F77=gfortran-
↪9 FC=gfortran-9

$ make all
$ make install
```

The following information is added in "~/.bash_profile" (or "~/.bashrc").

```
MPIROOT=$HOME/Software/mpi
export PATH=$MPIROOT/bin:$PATH
export LD_LIBRARY_PATH=$MPIROOT/lib:$LD_LIBRARY_PATH
export MANPATH=$MPIROOT/share/man:$MANPATH
```

Launch another terminal window or reload "~/.bash_profile" (or "~/.bashrc"):

```
$ source ~/.bash_profile
```

The OpenMPI tools should be installed in "$HOME/Software/mpi/bin".

```
$ which mpirun mpif90 mpicc
/Users/[username]/Software/mpi/bin/mpirun
/Users/[username]/Software/mpi/bin/mpif90
/Users/[username]/Software/mpi/bin/mpicc
```

Make sure that "mpicc" and "mpif90" are linked to "gcc-9" and "gfortran-9", respectively.

```
$ mpicc --version
gcc-9 (Homebrew GCC 9.2.0) 9.2.0

$ mpif90 --version
GNU Fortran (Homebrew GCC 9.2.0) 9.2.0
```

If you want to uninstall OpenMPI, just remove the directory "mpi" in "Software".

LAPACK/BLAS libraries

Finally, we install LAPACK and BLAS libraries. Again, "real" GNU compilers are used for the install.

```
$ cd $HOME/Software
$ wget http://www.netlib.org/lapack/lapack-3.8.0.tar.gz
$ tar -xvf lapack-3.8.0.tar.gz
$ cd lapack-3.8.0

$ cp make.inc.example make.inc
```

In the "make.inc" file, there are three lines to be modified:

```
#  CC is the C compiler, normally invoked with options CFLAGS.
#
CC     = gcc-9
...

#  should not compile LAPACK with flags such as -ffpe-trap=overflow.
#
FORTRAN = gfortran-9
...

#  load options for your machine.
#
LOADER   = gfortran-9
...
```

After the modification, we install BLAS and LAPACK libraries:

```
$ make blaslib
$ make lapacklib

$ ls lib*
liblapack.a  librefblas.a

$ ln -s librefblas.a ./libblas.a
```

The following information is added in "~/.bash_profile" (or "~/.bashrc").

```
export LAPACK_PATH=$HOME/Software/lapack-3.8.0
```

Launch another terminal window or reload "~/.bash_profile" (or "~/.bashrc"):

```
$ source ~/.bash_profile
```

If you want to uninstall LAPACK/BLAS, just remove the directory "lapack-3.8.0" in "Software".

BIBLIOGRAPHY

[1] D.A. Case, I.Y. Ben-Shalom, S.R. Brozell, D.S. Cerutti, T.E. Cheatham III, V.W.D. Cruzeiro, T.A. Darden, R.E. Duke, D. Ghoreishi, M.K. Gilson, H. Gohlke, A.W. Goetz, D. Greene, R Harris, N. Homeyer, S. Izadi, A. Kovalenko, T. Kurtzman, T.S. Lee, S. LeGrand, P. Li, C. Lin, J. Liu, T. Luchko, R. Luo, D.J. Mermelstein, K.M. Merz, Y. Miao, G. Monard, C. Nguyen, H. Nguyen, I. Omelyan, A. Onufriev, F. Pan, R. Qi, D.R. Roe, A. Roitberg, C. Sagui, S. Schott-Verdugo, J. Shen, C.L. Simmerling, J. Smith, R. Salomon-Ferrer, J. Swails, R.C. Walker, J. Wang, H. Wei, R.M. Wolf, X. Wu, L. Xiao, D.M. York, and P.A. Kollman. Amber18. University of California, San Francisco, 2018.

[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] J. Huang, S. Rauscher, G. Nawrocki, T. Ran, M. Feig, B. L. de Groot, H. Grubmüller, and A. D. MacKerell Jr. CHARMM36m: an improved force field for folded and intrinsically disordered proteins. *Nat. Methods*, 14:71–73, 2017.

[14] W. Humphrey, A. Dalke, and K. Schulten. VMD: Visual molecular dynamics. *J. Mol. Graph.*, 14:33–38, 1996.

[15] S. Jo, T. Kim, V. G. Iyer, and W. Im. CHARMM GUI: A web based graphical user interface for CHARMM. *J. Comput. Chem.*, 29:1859–1865, 2008.

[16] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *J. Am. Chem. Soc.*, 117:5179–5197, 1995.

[17] H. Taketomi, Y. Ueda, and N. Go. Studies on protein folding, unfolding and fluctuations by computer simulation. *nt. J. Peptide Proteins Res.*, 7:445–459, 1975.

[18] 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.

[19] 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.

[20] J. K. Noel, P. C. Whitford, K. Y. Sanbonmatsu, and J. N. Onuchic. SMOG@ctbp: simplified deployment of structure based models in GROMACS. *Nucleic Acids Res.*, 38:W657–W661, 2010.

[21] J. K. Noel, M. Levi, M. Raghunathan, H. Lammert, R. L. Hayes, J. N. Onuchic, and P. C. Whitford. SMOG 2: A Versatile Software Package for Generating Structure Based Models. *PLoS Comput. Biol.*, 12:e1004794, 2016.

[22] 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.

[23] 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.

[24] M. Feig, J. Karanicolas, and C. L. III Brooks. MMTSB Tool Set: enhanced sampling and multiscale modeling methods for applications in structural biology. *J. Mol. Graph, Model.*, 22:377–395, 2004.

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

[26] AMBER. http://ambermd.org/.

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

[28] 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.

[29] 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.

[30] 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.

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

[39] 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.

[40] 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.

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

[42] 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.

[43] 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.

[44] E. Braun, S. M. Moosavi, and S. Smit. Anomalous Effects of Velocity Rescaling Algorithms: The Flying Ice Cube Effect Revisited. *J. Chem. Theory Comput.*, 14:5262–5272, 2018.

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

[46] G. J. Martyna, M. L. Klein, and M. Tuckerman. Nose-Hoover Chains - the Canonical Ensemble Via Continuous Dynamics. *J. Chem. Phys.*, 97:2635–2643, 1992.

[47] G. J. Martyna, M. E. Tuckerman, D. J. Tobias, and D. J. Klein. Explicit reversible integrators for extended systems dynamics. *Mol. Phys.*, 87:1117, 1996.

[48] 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.

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

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

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

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

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

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

[55] 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.

[56] 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.

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

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] S. Re, H. Oshima, K. Kasahara, M. Kamiya, and Y. Sugita. Encounter complexes and hidden poses of kinase-inhibitor binding on the free-energy landscape. *Proc. Natl. Acad. Sci. USA*, 116:18404–18409, 2019.

[63] 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.

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

[65] 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.

[66] 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.

[67] Y. Miao, V. A. Feher, and J. A. McCammon. Gaussian Accelerated Molecular Dynamics: Unconstrained Enhanced Sampling and Free Energy Calculation. *J. Chem. Theory Comput.*, 11:3584–3595, 2015.

[68] Y. T. Pang, Y. Miao, Y. Wang, and J. A. McCammon. Gaussian Accelerated Molecular Dynamics in NAMD. *J. Chem. Theory Comput.*, 13:9–19, 2017.

[69] D. Hamelberg, J. Mongan, and J. A. McCammon. Accelerated Molecular Dynamics: A Promising and Efficient Simulation Method for Biomolecules. *J. Chem. Phys.*, 120:11919–11929, 2004.

[70] D. Hamelberg, C. A. F. de Oliveira, and J. A. McCammon. Sampling of Slow Diffusive Conformational Transitions with Accelerated Molecular Dynamics. *J. Chem. Phys.*, 127:155102, 2007.

[71] T. Shen and D. Hamelberg. A Statistical Analysis of the Precision of Reweighting-Based Simulations. *J. Chem. Phys.*, 129:034103, 2008.

[72] Y. Miao, W. Sinko, L. Pierce, D. Bucher, R. C. Walker, and J. A. McCammon. Improved Reweighting of Accelerated Molecular Dynamics Simulations for Free Energy Calculation. *J. Chem. Theory Comput.*, 10:2677–2689, 2014.

[73] H. Oshima, S. Re, and Y. Sugita. Replica-Exchange Umbrella Sampling Combined with Gaussian Accelerated Molecular Dynamics for Free-Energy Calculation of Biomolecules. *J. Chem. Theory Comput.*, 2019. URL: https://pubs.acs.org/doi/10.1021/acs.jctc.9b00761, doi:10.1021/acs.jctc.9b00761 (https://doi.org/10.1021/acs.jctc.9b00761).