MODYLAS Reference Manual

Version 1.0.2

4 Nov. 2014

Contents

1. Preface

The "MOlecular DYnamics Software for LArge Systems" (MODYLAS) is a general-purpose, molecular dynamics simulation program suited to the simulation of very large physical, chemical, and biological systems. MODYLAS supports most standard molecular dynamics calculation algorithms. In particular, for the calculations of the long-range Coulombic interaction, a combination of the fast multipole method (FMM) and the Ewald method for multipoles appropriate for the periodic boundary condition can be used. For temperature and pressure control, the Nosé-Hoover chain and Andersen method can be used respectively, to generate *NVT*, and *NPT* ensembles. For the numeric integration in solving the Newton's equations of motion, a multiple time-step algorithm called the rRESPA is being used. The distance constraints between atoms are treated using SHAKE, RATTLE, and ROLL algorithms. The program can handle all-atom force fields such as the CHARMM22 with CMAP and CHARMM36 with CMAP. In the near futre, the AMBER, and OPLSAA will also be supported. Further, the program will also support free-energy calculation algorithms based on the thermodynamic integration method.

The program also equips several newly developed methods to execute highly parallelized molecular dynamics calculations. The program ensures excellent scalability by the use of algorithms that practically eliminate data copying for communications and arithmetic operations, algorithms with minimal communication latency, and a parallel bucket-relay communication algorithm for the upper-level multipole moments in the FMM. Moreover, the use of blocked arithmetic operations can avoid the need to reload data from memory to cache, ensuring very low cache-miss rates. A benchmark test on MODYLAS using 65,536 nodes of the K-computer showed that the overall calculation time per step including communications is 5 ms for a 10 million atom system. This means that the simulation of a 10 million atom system can progress by 35 ns per day. MODYLAS thus enables us to study large-scale real systems such as viruses, liposomes, protein aggregates, micelles, and polymers.

2. License

MODYLAS Software Program License Agreement

MODYLAS Copyright Administrator (the "Licensor"), representing all the members listed in Exhibit 1 (the "Copyright Holders") who are the copyright holders of MODYLAS (the "Program"), grants to a person which is identified in the application page (the "Licensee") the right to use the Program in accordance with the terms and the conditions of this MODYLAS Software Program License Agreement (this "Agreement") free of charge.

Article 1 (Definition)
1. "Commercial use" means, including but not limited to, selling MODYLAS for a fee, doing consulting and contract calculation for a fee by using MODYLAS, using MODYLAS for business directly. "Non-commercial use" means the use of any other, including but not limited to, research and development in a for-profit company as well as academic use.

2. "Military use" means, including but not limited to, using MODYLAS to develop any military weapons and/or directly to other military actions. "Non-military use" of MODYLAS means the use of any other.

Article 2 (Provision of the Program)
1. The Program shall be provided for all the non-military, non-commercial research purposes (the "Purpose") only. If you would like to use MODYLAS for business, then please contact us at the address described in the end of this agreement.

2. The Licensor shall provide the Licensee with the Program in source code form.

3. The Licensor provides the Licensee with the reference manual, tutorial text, sample input and output relating to the Program (collectively called the "Manuals").

Article 3 (Granted Rights)
1. The Licensor grants to the Licensee the non-exclusive and non-transferable right to use the Program for the Purpose in accordance with the terms and the conditions of this Agreement. Under any circumstances the Licensee shall not use the Program for purposes other than the Purpose.

2. If the Licensee is a member or an employee of an organization for profit, the Licensee may, within the scope of the Purpose, use the Program and the Derivative Program (defined in

Paragraph 3 of this Article) for research and development for such organization in accordance with the provision of this Agreement.

3. The Licensee may modify the Program to the extent necessary for the Purpose. When a program as a derivative work (a "Derivative Program") is created as a result of modification of the Program by the Licensee, the Licensee shall promptly notify the Licensor of such fact. Upon request from the Licensor, the Licensee shall provide the Licensor with the Derivative Program in source code, and grant to the Licensor the non-exclusive right to use and modify the Derivative Program, and to distribute and license the Derivative Program to third parties without charging any fees (other than reimbursement of expenses for distribution) for the purpose of research.

4. Unless the Licensor grants a prior written consent, the Licensee shall not (i) provide to third parties or allow third parties to use the Program or any Derivative Program, (ii) make the Program or any Derivative Program public, or (iii) use the Program or any Derivative Program for any purpose other than the Purpose.

5. Regardless of the preceding paragraph, Licensee is permitted with other Licensee who is an employee or member of the same organization, to perform information exchange and technology transfer on MODYLAS, including Derived Programs.

6. The Licensee shall notify the Licensor of the Licensee's full legal name, the name of the organization to which the Licensor belongs, the Licensee's title in such organization and e-mail address. When any change is made for any item listed in the preceding sentence, the Licensee shall notify the Licensor of such change without delay.

7. The Licensee shall keep the Program and any Derivative Program confidential with utmost care, and shall be liable to the Licensor for any leak of information regarding the Program or any Derivative Program through negligence or willful misconduct of the Licensee.

Article 4 (Prohibition of the Use for the Purpose of Profit)

1. Unless upon prior written express consent from the Licensor, the Licensee shall not use the Program or any Derivative Program for commercial purposes other than research and development.

2. If the Licensee is a member or an employee of an organization with the purpose of profit, the Licensee's use of the Program or any Derivative Program for business purposes of such organization other than research and development shall fall under the use for commercial

purposes which is prohibited in the preceding Paragraph.

Article 5 (Retention of Title and Rights)
The Copyright Holders retain all the rights in the Program including ownership. The Program is protected by laws and treaties relating to copyright and other intellectual properties.

Article 6 (Provision of Information)
Upon request from the Licensor, the Licensee shall provide the Licensor with information relating to the use of the Program or any Derivative Program (including feedback, comments, bugs, etc.).

Article 7 (Prohibition of Use of Components of the Program)
The license under this Agreement is a license to use the Program as a single and inseparable program. Unless explicitly permitted in writing by the Licensor, the Licensee shall not separate any components of the Program (including the part in any Derivative Program derived from the Program, and the same shall apply hereinafter.) from the Program or incorporate any components of the Program into another program.

Article 8 (Term and Termination)
1. This Agreement shall be effective as of the time of the downloading any of MODYLAS program files and shall continue in effect unless terminated in accordance with this Article.

2. The Licensor may terminate this Agreement immediately when the Licensee makes a breach of this Agreement.

3. Each party may terminate this Agreement at any time by giving at least fourteen (14) days prior notice to the other party. If the notification by e-mail to the Licensee ends with an error, Licensor may notify Licensee by posting its contents on the homepage.

4. Upon termination of this Agreement, the Licensee shall stop using the Program and any Derivative Program entirely and shall destroy the Program, all Derivative Programs, their components and all of their duplications.

5. Article 3 Paragraph 3, Article 5, Article 6, Article 8 Paragraph 4, Article 9, Article 11, Article 12, Article 13, and Article 14 shall survive the termination of this Agreement.

Article 9 (Publication of the Results)
When the Licensee wishes to publish the results of the research using the Program or any

Derivative Program through theses or other means (each, a "Publication"), the Licensee shall include the following paper in reference.

> Andoh, Y. et al., "MODYLAS: A Highly Parallelized General-Purpose Molecular Dynamics Simulation Program for Large-Scale Systems with Long-Range Forces Calculated by Fast Multipole Method (FMM) and Highly Scalable Fine-Grained New Parallel Processing Algorithms," J. Chem. Theory Comput., 2013, 9 (7), pp 3201-3209.
> DOI: 10.1021/ct400203a

Notwithstanding the foregoing, if the main purpose of the Publication is to compare the computing performance of the Program with those of other programs, the Licensee shall consult with the Lisensor for such intention, one month or more before any benchmark test of the Program. The Licensee also shall obtain the Licensor's permission before submission of the Publication, including oral presentation.

Article 10 (Authority of the Licensor)
The Licensor hereby represents and warrants that it is given the authority to grant the right under this Agreement from Copyright Holders.

Article 11 (Entire Agreement)
This Agreement constitutes the entire agreement between the parties with respect to the subject matter hereof, and supersedes all prior written or oral agreements, understandings, proposals and representations with respect to the subject matter hereof.

Article 12 (Disclaimer)
The Licensor shall not be liable to the Licensee with respect to any damages that may arise in connection with the Program or the license regarding this Program, including but not limited to, defects in the Program, the use of the Program or inability to use the Program, even if the Licensor has been advised of the possibility of such damages.

Article 13 (Governing Law)
This Agreement shall be governed by and construed in accordance with the laws of Japan.

Article 14 (Jurisdiction)
Nagoya District Court, Japan shall have the exclusive jurisdiction over any dispute that may arise out of or in connection with this Agreement.

[Contact] MODYLAS Group
C/O Theoretical and Computational Chemistry Initiative (TCCI) Office
INSTITUTE FOR MOLECULAR SCIENCE (IMS)
Myodaiji, Okazaki, Aichi 444-8585, JAPAN
E-mail address: MODYLAS[at]yfep2.ims.ac.jp
FAX: 0564-54-2254
TEL: 0564-55-7074"

Exhibit 1　List of the Copyright Holders

Name	Organization & Title
*Susumu Okazaki,	Professor, Nagoya University
Noriyuki Yoshii,	Associate Professor, Nagoya University
Fumiyasu Mizutani,	Techinical Staff, Institute for Molecular Science
Kensuke Iwahashi,	Techinical Staff, Institute for Molecular Science
Atsushi Yamada,	Assistant Professor, Nagoya University
Yoshimichi Andoh,	Researcher, Nagoya University
Kazushi Fujimoto,	Assistant Professor, Ritsumeikan University
Hidekazu Kojima,	graduate student, Nagoya University

*: MODYLAS Copyright Administrator

3. Installation

3.1 Machine environment

- MODYLAS has been tested in the following systems:
  K-computer, FX10, and PC cluster
- Compiler
  The commands for compiling MODYLAS with MPI depend on the system. Therefore, you need to know the compiler installed in your system.
  - [Fujitsu] frtpx (e.g., mpifrtpx or mpifrt)
  - [Intel] ifort (e.g., mpif90 or mpiifort)
  - [PGI] pgf90 (e.g., mpif90)
- MPI library
  Normally, suitable MPI libraries are automatically linked by the compilation command for MPI.

3.2 Building MODYLAS

(A) Building modylas, modylas-text2bin, and modylas-mdtrj2xyz together

```
tar xvzf modylas-1.0.0.tar.gz
cd modylas-1.0.0/source/
setenv FC MPIF90CMD
```

Set FC environment variable only when the value set by the configure command is not suitable.

MPIF90CMD is the suitable Fortran 90 compiler for MPI.

```
setenv FCFLAGS FLAGS
```

Set FCFLAGS environment variable only when the value set by the configure command is not suitable.

In *FLAGS*, specify an optimization level.

```
./configure --with-kind-fortran-compiler=XXX
```

In *XXX*, specify the preset prepared for your system. The systems with preset are listed below. If your system is not listed below, specify whichever most closely corresponds to your system.

K	For the K-computer
FX10	For Fujitsu FX10
INTEL	For a system that uses an Intel compiler
PGI	For a system that uses a PGI compiler

After that, set the above environment variables as needed, and then execute the configure command again or directly edit src/Makefile.

```
cd src
make -j N
```

N is the number of jobs to compile simultaneously. When the -j option is omitted, source files are compiled one by one.

(B) Building only modylas-text2bin and modylas-mdtrj2xyz without MPI

```
tar xvzf modylas-1.0.0.tar.gz
cd modylas-1.0.0/source/
setenv FC F90CMD
```

Set FC environment variable only when the value set by the configure command is not suitable.

F90CMD is the suitable Fortran 90 compiler.

```
setenv FCFLAGS FLAGS
```

Set FCFLAGS environment variable only when the value set by the configure command is not suitable.

In this environment variable, you need to describe necessary settings, such as optimization level.

```
./configure --with-kind-fortran-compiler=XXX --disable-mpi
```

In *XXX*, specify the preset prepared for your system. Available systems are listed below. If your system is not listed, specify whichever most closely corresponds to your system.

K	For the K-computer
FX10	For Fujitsu FX10
INTEL	For a system that uses an Intel compiler
PGI	For a system that uses a PGI compiler

After that, set the above environment variable as needed, and then execute the configure command again or directly edit src/Makefile.

```
cd src
make clean
make -j N
```

N is the number of jobs to compile simultaneously. When the -j option is omitted, source files are compiled one by one.

4. Input/Output Files

All input/output files of MODYLAS must have a common name except for the extension. The common name is represented by "sessionname" in this manual. "sessionname" may be a user-specified string, but must not begin with a numerical character. Input/output files are distinguished from each other by the extension. The following section lists the input/output files:

4.1 List of input/output files

[Input files]

The “sessionname.mddef” file is used to define the details of the calculation conditions. The “sessionname.mdff” file is used to define force field parameters of each atom, and the “sessionname.mdxyz” file is used to define initial coordinates and velocities. As the input files, both ASCII-format and binary-format can be used. Normally, the use of ASCII-format input files is recommended. For the calculation of a very large system, however, the use of binary-format input files is recommended in order to reduce the time required to read the input files. If both ASCII-format and binary-format input files are found, the binary-format input files will override the ASCII-format input files.

- sessionname. mddef

  Calculation conditions (ASCII format)
- sessionname. mdff

  Force field information (ASCII format)
- sessionname. mdff.bin

  Binary-format version of sessionname.mdff (binary format)
- sessionname.mdxyz

  Atomic coordinates and velocities, and periodic-cell information (ASCII format)
- sessionname.mdxyz.bin

  Binary-format version of sessionname.mdxyz (binary format)
- sessionname.posiconst (option)

  External input file for specifying the constraint of atomic position (position constraint) (ASCII format). The constraint of atomic position can also be specified in the force field input file (sessionname.mdff). However, if the sessionname.posiconst file exists, the sessionname.posiconst file will override the force field input file, and, the specification in sessionname.mdff will be ignored.

  For the calculation of a very large system, the use of this external input file is strongly recommended, because MODYLAS execution module takes long time to read the position constraint information in the sessionname.mdff.

  When sessionname.mdff.bin is used instead of sessionname.mdff, this external input file must be created because any position constraint information are not converted into sessionname.mdff.bin.

(*) All the binary-format files must use the same endian format. When running MODYLAS on the K-computer with binary inputs in the little-endian format, “FORT90L='-Wl,-T'” must be specified in environment variable.

[Output files]

- sessionname.mdrun

  Information on the time spent for the calculation (ASCII format)

- sessionname.mdmntr

  Monitored values of physical quantities such as the Hamiltonian, temperature, and pressure for each time step (ASCII format)

- sessionname.mdtrj.bin

  Trajectory (atomic coordinates and velocities) information for analysis (binary format)

- sessionname.restart.bin

  Atomic Coordinates and velocities at the last time step for restart (binary format). This file has the same format as the input file of sessionname.mdxyz.bin.

- sessionname.restart.asc

  Atomic coordinates and velocities at the last time step for restart (ASCII format), which are in the same format as the input file of sessionname.mdxyz. This file is created by default, but is not created if "ascii=no" is specified between the <output> tags in sessionname.mddef.

(Note) Trajectory information file in the DCD format (i.e., sessionname.dcd) will be supported in near future.

5. Preparation of Input Files

5.1 Overview of input file format

Keywords and parameters are specified in tag format in input files. Each tag is opened with <tag> and closed with </tag>. The tag types are specified by character string "tag" within <…>. The tag name, "tag", reflects the meaning of the keywords or parameters. Tags have hierarchical structure. Then, do not change the relationship between parent and child tags, that is, the order of tags within the hierarchy. In the case of a single keyword or parameter value, the value is given by format of "variable name=value" between open and close tags. When there are two or more values like coordinates and other data, separate the individual values in a tag with either a space or a return. The variable names that can be used are fixed depending on the parent tag. There are three types of the values: integer, real, and character. These can be separated by a space or return, but a tab is not allowed. A simple example of input files is located in the downloaded source file as sample input files (see the Tutorial). All of the input keywords and options are listed in the Appendix.

5.2 How to create an input file

The following software supports the creation of input files for MODYLAS.

- Input generator for molecular science: Nano-Ignition (Free software, web site: http://nano-ignition.ims.ac.jp/). This software can build or edit molecular system and set its force field parameters after reading atomic coordinates from a PDB-formatted file, then, input files of MODYLAS (or other software) are generated.
- VMD software (website: http://www.ks.uiuc.edu/Research/vmd/) with a plug-in program for MODYLAS (to be distributed in near feature) for extended functions can be used. This software can output the molecular system information generated by VMD in the input file format of MODYLAS.
- A software to be distributed in near feature can create the calculation condition input file (sessionname.mddef) for MODYLAS

5.3 Converting input files into binary format

You can use the conversion program, modylas-text2bin, compiled under the source/src/ directory to convert ASCII-formatted input files "sessionname.mdff" and "sessionname.mdxyz" into binary-formatted input files "sessionname.mdff.bin" and "sessionname.mdxyz.bin," respectively.

```
%>  (path)/modylas-text2bin    sessionname
```

(Note) When you use binary-format input file, the atomic position constraint keyword (keyword specified between the <position constrain> tags in the force field input file) is invalid. If you want to set position constraint by using a binary-format input file, use external input file, sessionname.posiconst file.

6. How to Run MODYLAS

6.1 Run procedure

This section describes how to run MODYLAS.

(1) Prepare the input files (see the previous chapter).

(2) Copy execution module, "modylas", generated by compilation into the directory that contains a set of input files, or link the module to that directory.

(3) Create a job script file according to the machine environment of your system, and submit the job (see the manual for the machine environment to be used).

As an example, the following shows how to submit a job on the K-computer. After creating a job script file (described later), execute the following command to submit the job,

```
%> pjsub   ./run_K.sh
```

"pjsub" is the command to submit jobs into the K-computer. "run_K.sh" is the job script file created by the user, in which parallelization environment information, such as, file names and the number of nodes, are specified.

To check the status of a job, use the following command:

```
%>  pjstat
```

To delete a job, use the following command:

```
%>  pjdel   (job-id-number)   (Use the pjstat command to confirm the ID number of a job.)
```

If you want to execute a calculation directly from the command line instead of submiting a batch job, use the following command:

```
%>  mpiexec -n 8 (path)/modylas   sessionname
```

[Job script file (when using the K-computer)]

A job script file ("run_K.sh") is shown on the next page. The script is also included in the directory for sample input files, and can be copied and modified for use. For details on the options, see the distributed manual for the K-computer. Here, a simple explanation is given for some important options that users often modify (① to ⑧ on the next page).

① Number of nodes. Now, only the power-of-two number (2^m and $m \geq 3$) is available in the current version

② Upper limit on the computation time

③ Job class according to the number of nodes: "micro", "small", "large", "huge" or "huge2".

④ Stage-in option, which specifies the file names and paths to be used to upload the input files to the K-computer.

(For each binary-format input file, change the file name by adding extension ".bin".)

⑤ Stage-out option, which specifies the file names and paths to be used to download the output files from the K-computer.

⑥ Language environment, which must be same as the language environment when MODYLAS execution module was compiled.

⑦ Number of MPI processes (same to ①)

⑧ Specification of MODYLAS execution module and session name of input files. Write "sessionname" after "modylas".

```
#!/bin/sh
#PJM --rsc-list "node=16"                    ←①
#PJM --rsc-list "elapse=0:10:00"             ←②
#PJM --rsc-list="rscgrp=small"               ←③
#PJM --stg-transfiles all
#PJM --mpi "use-rankdir"
#PJM --stgin    "rank=* ./modylas                %r:./"      ←④
#PJM --stgin    "rank=* ./sessionname.mddef      %r:./"
#PJM --stgin    "rank=* ./sessionname.mdff       %r:./"
#PJM --stgin    "rank=* ./sessionname.mdxyz      %r:./"
##
#PJM --stgout "rank=0 %r:./sessionname.mdmntr      ./ "   ←⑤
#PJM --stgout "rank=0 %r:./sessionname.mdrun       ./ "
#PJM --stgout "rank=0 %r:./sessionname.restart.bin ./ "
#PJM --stgout "rank=0 %r:./sessionname.restart.asc ./ "
#PJM --stgout "rank=0 %r:./sessionname.mdtrj.bin   ./ "
#PJM -s
#-------------------------------------------------------------------------
# path
. /work/system/Env_base_1.2.0-15          ←⑥

# Parameters
NPROCS=16                          ←⑦
NTHREADS=8
export PARALLEL=${NTHREADS}
export OMP_NUM_THREADS=${NTHREADS}
LPG="/opt/FJSVxosmmm/sbin/lpgparm -t 4MB -s 4MB -h 4MB -d 4MB -p 4MB"

# Endian
export FORT90L='-Wl,-T'

# MCA Parameters
MCA_PARAM="--mca common_tofu_fastmode_threshold 0"
MCA_PARAM="${MCA_PARAM}  --mca  common_tofu_max_fastmode_procs  40  --mca
common_tofu_large_recv_buf_size 2097152"
#-------------------------------------------------------------------------
pwd

date
LD="./modylas   ./sessionname "          ←⑧
mpiexec -n ${NPROCS} ${MCA_PARAM} ${LPG} ${LD}
date
```

6.2 Restart procedure

To continue a calculation after a last calculation of MODYLAS, prepare the set of input files described in Section 4.1. For example, to continue the calculation by generating the input files with session name "sessionname2" from the calculation with session name "sessionname1,"

prepare input file "sessionname2," as described below.

- Calculation condition file:
  Copy the input file used for the previous calculation.
  ```
  %> cp  sessionname1.mddef   sessionname2.mddef
  ```
  If necessary, modify the calculation conditions in the input file for restart.
- Force field file:
  [ASCII format] `%> cp  sessionname1.mdff    sessionname2.mdff`
  [Binary format] `%> cp  sessionname1.mdff.bin  sessionname2.mdff.bin`
- Coordinate and velocity file:
  [ASCII format] `%> cp  sessionname1.restart.asc  sessionname2.mdxyz`
  [Binary format] `%> cp  sessionname1.restart.bin  sessionname2.mdxyz.bin`
- Option file:
  ```
  %> cp  sessionname1.posiconst   sessionname2.posiconst
  ```
  These files may be linked instead of being copied (to link the file, replace the cp command with "ln -s".)

To restart the calculation, replace each input file name with "sessionname2" (in the job script file or command line), and then submit a calculation job or execute a calculation command in the same way as described in the previous section.

7. Monitor of Calculation Results

7.1 Output of physical quantities

Physical quantities such as Hamiltonian, temperature, and pressure values are written to the sessionname.mdmntr file sequentially at the step intervals specified in the interval variable between the <monitor> tags in the sessionname.mddef file.

[sessionname.mdmntr output sample]

```
## n0101.mdmntr -- monitor variables output from MD calculation by modylas
#
# datas below are formated as:
# step        time                Hamiltonian          potential-E          kinetic-E          total energy
temperature          volume                pressure          box-length(x)          box-length(y)
box-length(z)
#                        [sec]          [J/cell]          [J/cell]          [J/cell]          [J/cell]
[K]                [m3]                [Pa]                [m]                [m]
[m]
#
          1   2.000000000000E-15 -1.080596504280E-16 -2.464759127634E-16   1.305735314800E-16
-1.159023812835E-16   3.260776624164E+02   2.491621263963E-25   6.596619488522E+06
5.759307554640E-09   5.490470367063E-09   8.586602898096E-09
          2   4.000000000000E-15 -1.080970356668E-16 -2.469468756070E-16   1.310090310261E-16
-1.159378445810E-16   3.271652233668E+02   2.491678749128E-25   5.577704315386E+06
5.759411245174E-09   5.490486488292E-09   8.586695250716E-09
```

In the sessionname.mdmntr file, header lines (beginning with #) indicate the names of the output physical quantities (on the fourth line) and the units of the physical quantities (on the fifth line). The contents of the header lines are described below.

1.	step	Time step number In the case of multiple time-step algorithm, the number corresponds to the step for the long-range interaction.		
2.	time	time (= dt*step)	[sec]	
3.	Hamiltonian	Hamiltonian for the system (particle system + heat bath + pressure bath) (Heat bath for NVT ensemble, pressure bath for NPT ensemble)	[J/cell]	
4.	potential-E	Potential energy of particle system (intra- + inter- moleclar potential energy)	[J/cell]	
5.	kinetic-E	Kinetic energy of particle system	[J/cell]	
6.	total energy	Total energy of particle system: potential-E + kinetic-E	[J/cell]	
7.	temperature	Temperature of particle system	[K]	
8.	volume	Volume of unit cell	[m^3]	
9.	pressure	Bulk pressure of system: (Pxx + Pyy + Pzz)/3	[Pa]	
10.	box-length(x)	Absolute value of unit cell vector ***a***		[m]
11.	box-length(y)	Absolute value of unit cell vector ***b***		[m]
12.	box-length(z)	Absolute value of unit cell vector ***c***	[m]	

The physical quantities at each step are printed after the sixth line. These values can be graphed by graph display software.

For example, the graph of values can easily be displayed using gnuplot. In an environment in which gnuplot is installed, type the following command to start gnuplot:

```
%> gnuplot
```

You can display a desired graph on the screen by typing commands after "gnuplot>". For example, to see the time course of Hamilton, type as following,

```
gnuplot> plot 'sessionname.mdmntr' using 2:3 with lines
```

Here, the string between single quotations (' ') specifies the file name, "X:Y" after "using" specifies display of the data in column Y for the data in column X, and "with lines" specifies the connection of displayed data by line.

[sessionname.mdrun output sample]

```
## n0101.mdrun -- run time information of MD calculation by modylas
#
step:            12500
CPU time:        6086.753741 [sec]
for MD:          6084.206923 [sec]
time/step:          0.486737 [sec/step]
```

The contents of the sessionname.mdrun file are as follows:

- 1st and 2nd lines: Header part (beginning with #)
- 3rd line and after:
  - ・Number of elapsed steps
  - ・Total wall time of calculation by MODYLAS
  - ・Elapsed total time of MD calculation, excluding I/O processing
  - ・time of MD calculation per step

Because the sessionname.mdrun file is updated at every step, this file helps us to know the number of calculated time-steps that have been completed after starting the job, the number of time steps at which the job ends abnormally, and other information. The file also enables us to roughly estimate the number of steps performed per day by dividing the number of seconds in a day (86,400) by the time per step.

7.2 Convert trajectory data into a coordinate file

The sessionname.mdtrj.bin file has the output data that is the most important to research, which contains data of the trajectories (coordinates and velocities) of all the atoms as functions of time. This output file cannot be read directly by a text editor because it is a binary-format file. Then, auxiliary program "modylas-mdtrj2xyz" is available to convert sessionname.mdtrj.bin into a xyz-format file in ASCII format. To use modylas-mdtrj2xyz, first move to the folder that

contains the set of input files and the output .mdtrj.bin file, and then type the following:

```
%>  ./ modylas-mdtrj2xyz   sessionname
```

How many atoms to display?

(if negative value is given, all atoms are used)

26135 (Number of atoms you want to convert)

How many interval steps to display?

1 (Output interval, different from the "interval" in the sessionname.mddef file)

After executing, modylas-mdtrj2xyz generates a sessionname.xyz file having the following format in the current folder:

```
      26135                                                          <- # of atoms
 #      12500                                                        <- # of step
N   0.1687303373E+02   0.2334932507E+01   0.1559506824E+02          <- Atom names,
C   0.1762886632E+02   0.3602581521E+01   0.1560530104E+02          coordinates (x, y, z)
...
O   0.1328233085E+02  -0.1553688180E+02  -0.1786405457E+02
H   0.1416506470E+02  -0.1588953921E+02  -0.1775160214E+02
H   0.1339235805E+02  -0.1480280702E+02  -0.1846842207E+02
```

The atomic coordinates of the system are printed in succession and in chronological order. You can use this xyz file for analysis, and also use for viewing its structure by using visualization software, e.g., VMD or rasmol.

8. Parallelization of MODYLAS

8.1 Implementation of parallelization

Parallelization has been implemented in MODYLAS at the following three levels:

・MPI (Number of processes "NPROCS" is limited to 2^m and $m \geq 3$ in ver.1.0.0.)

・OpenMP

・SIMD

Basically, the parallelization of MODYLAS is premised on parallelizing by MPI between nodes, by OpenMP between cores in a node, and by SIMD between vector computing units in a core. You can also configure two MPI processes in a node and reduce the number of parallel threads by half, if necessary.

The MPI parallelization in MODYLAS is optimized in a way that minimizes communication time in a three-dimensional torus network. The three-dimensional torus network is a communication network that deploys nodes in the *x*, *y*, and *z* directions in the shape of a rectangular parallelepiped and enables internode communication only between the directly neighboring nodes in the *x*, *y*, or *z* direction (Figure 8-1). Therefore, to maximize the MPI-parallelized performance of MODYLAS, the three-dimensional shape of the nodes specified by the user must conform to the three-dimensional shape of each process to which a cell block (Section 8.2) created by domain decomposition is assigned. If, for example, a three-dimensional shape of 16 x 8 x 8 is selected for the nodes, "ndivx=16," "ndivy=8," and "ndivz=8" (total of 1024 processes) must be specified for the process shape. Appendix B gives examples of the node shape and process shape specifications.

Calculation can be executed even if the node shape and process shape are not completely consistent, but, in such a case, the optimization of parallelization for communication time becomes less effective. The user must determine whether to specify a node shape on a case by case basis, while considering both the increase in the job queuing time due to the specification of the node shape and the increase in the computing speed.

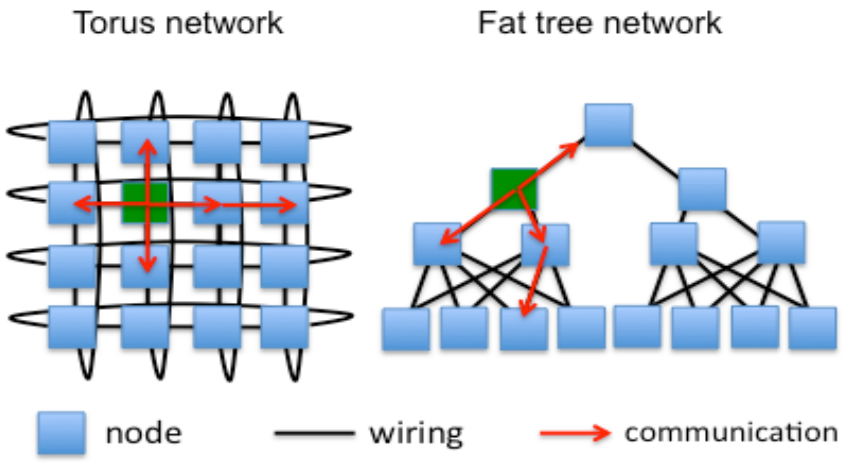

Figure 8-1 A schematic of the torus network and fat-tree network

Regarding thread parallelization using OpenMP, an explicit directive is inserted into each do loop that requires a large amount of calculation by the program (hot spot). For the other do loops, a compiler option is specified in Makefile to parallelize threads using the automatic parallelization feature of the compiler. The number of threads can be set to 1, 2, 4, 8, and 16 according to the number of cores in a node (core number greater than 16 is not checked, but would work). The number of threads in the parallelized do loops by the insertion of the directive is specified in environment variable "OMP_NUM_THREADS". The number of threads in the automatically parallelized do loops are specified in environment variable "PARALLEL". The values of these two variables must always match unless there is some particular reason to do otherwise.

Regarding SIMD, program coding is intended to eliminate "if" statements and extend the length of loops so that the automatic SIMD parallelization by the compiler can provide high performance at hot spots. Performance is optimized for the use of the Fujitsu compiler "frtpx".

8.2 Decomposition of space domain

In MPI parallelization, a MPI process performes operations related to the particles in a domain that is assigned to the process by dividing the unit cell in a grid pattern by domain decomposition.

The domains are assigned to individual processes as follows (Figure 8-2). First, each side of the cubic unit cell is equally divided by the number specified in input parameter "ncell" (= 2^n). The equally divided small domains are called "subcells." The total number of subcells is 2^{3*n}. Note that there is a restriction on the length of a side of a subcell, such that the length is not less than half of the cutoff radius of the Lennard-Jones potential. Next, processes are allocated to individual sides *x*, *y*, and *z* according to the settings of input parameters "ndivx," "ndivy," and "ndivz." As a result, 2^{3*n-m} subcells are equally distributed to each process (here, ndivx*ndivy*ndivz = 2^m). The group of subcells distributed to a process is called a "cell block."

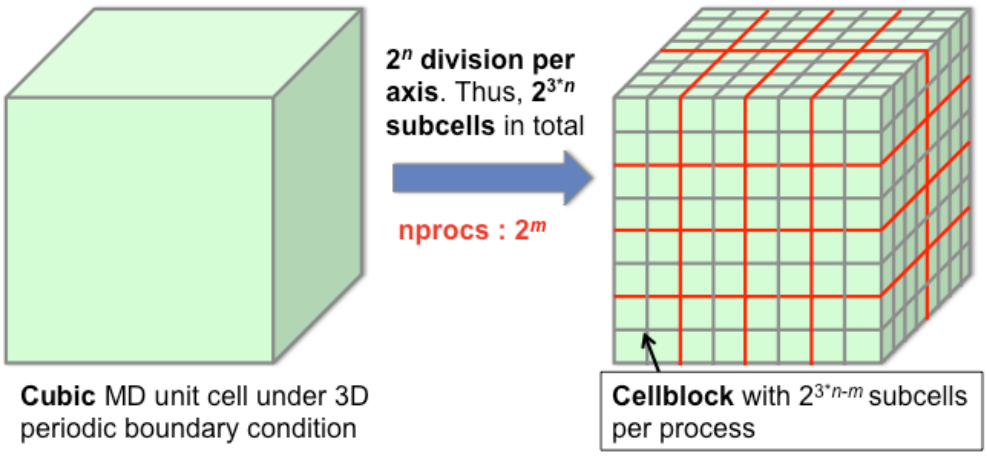

Figure 8-2 Decomposition of cubic unit cell into subcells and assignment of subcells to processes

When the three-dimensional shape of the node to be used matches the three-dimensional shape of the process, the following description may be useful to improve MPI parallelization efficiency. From the standpoint of communication load balancing, each cell block should be a cubic shape. If each cell block is a rectangular parallelepiped, parallelization performance is expected to be improved when the rectangular parallelepiped cell block is designed to be thin in *x* direction and thick in *z* direction. This is because data cumulative communication is performed in the order of *z*, *y*, and *x* directions in the program.

9. [Appendix] Input Keywords and Options

For input files, the method of entering input values must follow the tag format described below. When there is only a single input value for a parameter, the "key=value" format may be used. When there are multiple input values for a parameter, a space and linefeed code may be entered between the input values, excepting that a tab code must not be inserted between the input values. Notice that the serial numbers of atoms and molecules begin with 0.

```
(Example)
<atom>
  natom  = 3
  <positions>
  0.0    0.0    0.0
  1.0    0.0    0.0
  0.0    1.0    0.0
  </positions>
</atom>
```

In the above example, values are set for parameters "natom" and "positions" between the <atom> tags. This specification means that there are three atoms (natom = 3) and that their positions (position coordinates) are (0.0 0.0 0.0), (1.0 0.0 0.0), and (0.0 1.0 0.0).

Note that the following notation is hereafter used to indicate that a parameter (natom or positions in this example) is specified between the <atom> tags.

'/atom/natom'

'/atom/positions'

In the following three sections (9.1, 9.2, and 9.3), each keyword is explained in the commom format: A word indicating the type of variable, "String", "Integer", or "Real", multiplied by number of input values with "×" in the parentheses, e.g. (String × 1), is followed by its legend.

9.1 Calculation keywords (.mddef)

/input/version

(String × 1) Version number of the format of input files. The default is "1.0.0". Input files in the older format are available by using the keyword of "0.9.0b".

/output/ascii

(String × 1) Flag to output the file for restart in ASCII format. Specify "no", when the file is not to be output in ASCII format. When the specification of this keyword is omitted, an

ASCII-format file is output (default).

/output/trajectory/start

(Integer × 1) Number of the step at which trajectory file (.mdtrj.bin) starts to be output. Out put of the trajectory file is skipped until step number reaches the specified step number.

/output/trajectory/interval

(Integer × 1) Number of steps that specifies the interval of trajectory file output

/output/restart/start

(Integer × 1) Number of the step at which restart file (.restart.asc, or .restart.bin) starts to be output. Normally, specify 0.

/output/restart/interval

(Integer × 1) Number of steps that specifies the interval of restart file output. The file for restart is output by overwriting the previously output file for restart.

/output/monitor/start

(Integer × 1) Number of the step at which the file to monitor thermodynamic quantities (.mdmntr) starts to be output. Normally, specify 0.

/output/monitor/interval

(Integer × 1) Number of steps that specifies the interval of monitor file output

/integrator/dt

(Real number × 1) Time step of MD calculation

<Unit> sec

/integrator/steps

(Integer × 1) Number of steps in MD calculation

/integrator/multiple time step/nstep_skip_middle

(Integer × 1) Number of calculations for short-range interactions (interactions in a molecule) per calculation of middle-range interaction (LJ interaction and direct Coulombic interaction part of FMM). The default is 1.

/integrator/multiple time step/nstep_skip_long

(Integer × 1) Number of calculations for middle-range interactions per calculation of long-range interaction (multipole interaction part of FMM). The default is 1.

/integrator/optimize/step_length

(Real number × 1) Step length of structural optimization. The default is 0.2.

<Unit> Å

/integrator/optimize/convergence_condition (No effect in this version)

(String × 1) Physical quantity to determine the convergence of structural optimization (opt keyword)

"max_force" … Largest force among all atoms (default)

"mean_force" … Mean force among all atoms

"d_p_energy" … Change of potential energy per step

/integrator/optimize/convergence (No effect in this version)

(Real number × 1) Condition for the convergence of structural optimization, or the threshold for the physical quantity specified in the convergence_condition keyword

<Unit> kg m/s^2 for "max_force" and "mean_force" or J for "d_p_energy"

/integrator/optimize/up_rate (No effect in this version)

(Real number × 1) Rate of increase of the step length of structural optimization. The default is 1.0.

/integrator/optimize/down_rate (No effect in this version)

(Real number × 1) Rate of reduction of the step length of structural optimization. The default is 1.0.

/integrator/shake/maxiteration

(Integer × 1) Maximum number of iterative calculations of P-SHAKE. If the number of iteration exceeds this value, the program is forcibly terminated. The default is 1000.

/integrator/shake/shake_tolerance

(Real number × 1) Threshold for the convergence of P-SHAKE. The value is also used as the threshold for the convergence of P-RATTLE. The default is 10^{-10}.

<Unit> No dimension

/ensemble/ensemble

(String × 1) Type of ensemble: Specify one of the following types.

"opt" (structural optimization: steepest descent method)

"nve" (microcanonical ensemble)

"nvt" (canonical ensemble)

"npt_a" (isothermal-isobaric ensemble of non-rhombic unit cells)

"npt_pr" (isothermal-isobaric ensemble of rhombic cells based on Parrinello-Rahman method) (No effect in this version)

/ensemble/temperature

(Real number × 1) Temperature to be set for isothermal ensemble (nvt or npt_a), and velocity scaling

<Unit> K

/ensemble/maxwell_velocities

(String × 1) Specify "yes" when giving the initial velocities as a Maxwell-Boltzmann distribution.

/ensemble /velocity_scaling

(String × 1) Specify "yes" when scaling velocities to realize the set temperature in nve ensemble.

/ensemble/thermostat/tau_Q

(Real number × 1) Value of temperature control parameter "τ_Q" in the Nosé-Hoover chain

method. This value is required when the ensemble is "nvt," "npt-a," or "npt_pr" (currently not supported).

<Unit> sec

/ensemble/thermostat/initialize

(String × 1) Specify "yes" when clearing the variables for the heat bath position and momentum at the beginning of calculation.

/ensemble/pressure

(Real number × 1) Pressure to be set for isobaric ensemble (npt_a)

<Unit> Pa

/ensemble/Pref_inner

(Real number × 1) Value to be set as the partial pressure derived from a short-range interaction (interaction in a molecule)

<Unit> Pa

/ensemble/Pref_outer

(Real number × 1) Value to be set as the partial pressure derived from a middle-range interaction (LJ interaction and direct Coulombic interaction part of FMM)

<Unit> Pa

/ensemble/Pref_outermost

(Real number × 1) Value to be set as the partial pressure derived from a long-range interaction (multipole interaction part of FMM)

<Unit> Pa

/ensemble/barostat/tau_Q

(Real number × 1) Value of pressure control parameter "τ_Q" in the Andersen method

This value is required when the ensemble is "npt-a," or "npt_pr" (not supported by this version).

<Unit> sec

/ensemble/barostat/tau_W

(Real number × 1) Value of pressure control parameter "τ_W"

This value is required when the ensemble is "npt-a," or "npt_pr" (not supported by this version).

<Unit> sec

/ensemble/barostat/initialize

(String × 1) Specify "yes" when clearing the variables for the heat bath position and momentum at the beginning of calculation.

/intermolecular interaction/twobody/cutoff'

(Real number × 1) Cutoff distance for Lennard-Jones interaction

<Unit> Å

/intermolecular interaction/twobody/LJcorrection

(String × 1) Flag for whether to include LJ correction terms ("yes") or not ("no").

/intermolecular interaction/type

(String × 1) Type of periodic boundary condition. Specify "FMM" when using the FMM to calculate the Coulombic interaction under periodic boundary condition.

/intermolecular interaction/ncell

(Integer × 1) Number of divisions of domain in each of the x, y, and z directions in the calculation system using FMM.

/intermolecular interaction/fmm/ULswitch

(Integer × 1) Timing to switch the method of multipole information communication in FMM. Optimal value is 0, 1, 2, 3 when ncell=8, 16, 32, 64, respectively.

/intermolecular interaction/fmm/sterm

(String × 1) Flag for whether to include the contribution of the Ewald surface term ("yes") or not ("no"). The default is "no".

/intermolecular interaction/fmm/nmax

(Integer × 1) Order of multipole expansion. The default is 4.

/mpi/division

(String × 1) Specify "auto", when the numbers of processes to be allocated in the x, y, and z directions are automatically set. To manually set the numbers of processes, specify "manual", and then enter values for "nxdiv," "nydiv," and "nzdiv," below.

/mpi/nxdiv

(Integer × 1) Number of processes to be allocated in the x direction. The number must be a power of two in this version. The value of "nxdiv × nydiv × nzdiv" must be equal to the total number of processes.

/mpi/nydiv

(Integer × 1) Number of processes to be allocated in the y direction. The number must be a power of two in this version. The value of "nxdiv × nydiv × nzdiv" must be equal to the total number of processes.

/mpi/nzdiv

(Integer × 1) Number of processes to be allocated in the z direction. The number must be a power of two in this version. The value of "nxdiv × nydiv × nzdiv" must be equal to the total number of processes.

9.2 Force fields (.mdff)

/forcefield/type

(String × 1) Type of force field. One of three strings, such as "charmm," "oplsaa," and "amber" can be entered.

(*) Note, however, that this MODYLAS version (1.0.0) supports only "charmm".

/forcefield/cmap version

(String × 1) Version of CMAP to be specified when "charmm" is specified as the type of force field. "22" or "36" can be specified. The default is 36.

/system

([Integer 1, integer 2] × [number of molecule types])

"Integer 1" must be an ID number that indicates a type of molecule. Assign ID numbers, beginning from 0, to the molecule types.

"Integer 2" must be the number of molecules of the identified type.

/position constrain/type

(String × 1) Type of position constraint. One of the following three strings can be entered:

"harmonic" (Constraint that binds atomic positions to absolute coordinates by harmonic springs. This type of constraint can be used for all ensembles and structural optimization.)

"huge_mass" (Constraint that hugely increases the atomic mass. This type of constraint can be combined with SHAKE. This type of constraint can be used for NVE and NVT ensembles. All the atoms that have the same ID number as that specified in "/position constrain/atom/molecule," below are constrained.)

"fix" (Constraint that can be used only for structural optimization. This type of constraint fixes atomic coordinates to input values.)

/position constrain/force_constant

(Real number × 1) Spring constant to be specified when the type of position constraint is "harmonic".

<Unit> kcal/(mol Å^2)

/position constrain/atom/atom_type

(String × 1) When a value is specified for "/position constrain/atom/molecules," specify the type(s) of atoms on which to impose constraint conditions.

"all" (all types of atoms) or "heavy_atoms" (only heavy atoms other than hydrogen) can be specified.

/position constrain/atom/molecules

(String × 1) Specification of the molecule(s) to be constrained. "all" or "specified" can be specified.

When "specified" is choosen, values must be specified for "/position

constrain/atom/nmolecule" and "/position constrain/atom/molecule," below.

/position constrain/atom/nspecies

(Integer × 1) Number of molecules subject to position constraint

/position constrain/atom/species

([Integer 1, integer 2] × [number of molecules subject to position constraint])

"Integer 1" must be an ID number that indicates a type of molecule. "Integer 2" must be the ID number of a molecule (within the type of molecule).

/position constrain/position/type

(String × 1) Setting of the coordinates to be used for constraint

"initial" (initial coordinates) or "specify" (specified coordinates) can be specified.

When "specified" is choosen, values must be specified for "/position constrain/position/coordinate," below.

/position constrain/position/coordinate

([Integer, real number 1, real number 2, real number 3] × [number of atoms to be constrained])

"Integer" must be the number given to the atom for which "specified" coordinates are to be set.

"Real number 1," "real number 2," and "real number 3" must be the values of the specified x, y, and z coordinates, respectively.

<Unit> Real number 1: Å, real number 2: Å, real number 3: Å

/position constrain/atom/natom_allow_unconstrain

(Integer × 1) Number of the atoms that belong to the molecule to be constrained but are not subjected to position constraint

This specification overrides other position constraint settings.

/position constrain/atom/allow atom unconstrain

(Integer × number of atoms that belong to the molecule to be constrained but which are not subject to position constraint)

ID number of the atom, within the whole system, that belongs to the molecule to be constrained but which is not subject to position constraint

/topology and parameters/nmolecule

(Integer × 1) Number of types of molecules included in the system. This is not the total number of molecules.

/topology and parameters/molecule

Tag for specifying the details of an individual type of molecule. Write this tag in parallel as many times as the number of types of molecules.

/topology and parameters/species/id

(Integer × 1) ID number of the type of a molecule

/topology and parameters/species/natom

(Integer × 1) Number of atoms that constitute a specific molecule

/topology and parameters/species/segments/nsegment

(Integer × 1) Number of molecular segments

For convenience, MODYLAS refers to the clumps that are grouped in the topology file of CHARMM as "segments."

/topology and parameters/species/segments/segment

Tag for specifying the details of individual segments. Write this tag in parallel as many times as the number of segments.

/topology and parameters/species/segments/segment/ID

(Integer × 1) ID number of a segment. Assign ID numbers, beginning from 0, to the segments.

/topology and parameters/species/segments/segment/natom

(Integer × 1) Number of atoms included in the segment

/topology and parameters/species/segment/atom

(Integer × number of atoms included in the segment) Intramolecular numbers given to the atoms included in the segment

The intramolecular numbers must begin from 0.

/topology and parameters/species/mass

(Real number × number of atoms of a specific molecule) List the atomic weights of the atoms belonging to the molecule in the alignment sequence of atoms.

<Unit> g/mol

/topology and parameters/species/charge

(Real number × number of atoms of a specific molecule) List the atomic charges of the atoms belonging to the molecule in the alignment sequence of atoms.

<Unit> coulomb(C)

/topology and parameters/species/nvoidpair_coulomb

(Integer × 1) Number of coulomb void pairs in a molecule

/topology and parameters/species/coulomb void pair

([Integer 1, integer 2] × [number of coulomb void pairs in a molecule])

"Integer 1" and "integer 2" must be the intramolecular numbers given to the atoms that form a coulomb void pair.

/topology and parameters/species/nspecialpair_coulomb

(Integer × 1) Number of coulomb special pairs in a molecule

/molecules/molecule/coulomb special pair (* This specification is not used for CHARMM.)

([Integer 1, integer 2, real number] × [number of coulomb special pairs in a molecule])

"Integer 1" and "integer 2" must be the intramolecular numbers given to the atoms that form a coulomb special pair.

"Real number" must be the charge of the coulomb special pair.

<Unit> Real number: coulomb(C)

/topology and parameters/species/epsilon

([Real number] × [number of atoms]) LJ parameter εof the atoms that belong to a molecule

<Unit> kcal/mol

/topology and parameters/species/r

([Real number] × [number of atoms]) LJ parameter R_0 of the atoms that belong to a molecule

<Unit> Å

/topology and parameters/species/nvoidpair_lj

(Integer×1) Number of LJ void pairs in a molecule

/topology and parameters/species/lj void pair

([Integer 1, integer 2] × [number of LJ void molecular pairs]) Write the intramolecular numbers given to a set of atoms that form an LJ void pair, [Integer 1, integer 2], as many times as the number of LJ void pairs.

/topology and parameters/species/nspecialpair_lj

(Integer ×1) "Integer" must be the number of LJ special voids in a molecule.

/topology and parameters/species/lj special pair

([Integer 1, integer 2, real number 1, real number 2] × [number of LJ special molecular pairs])

"Integer 1" and "integer 2" must be the intramolecular numbers of the atoms that form an LJ special molecular pair.

"Real number 1" must be the ε_{14} value for the LJ special molecular pair.

"Real number 2" must be the R_{14} value for the LJ special molecular pair

Write the above setting as many times as the number of LJ special molecular pairs.

<Unit> Integer 1: none, integer 2: none, real number 1: kcal/mol, real number 2: Å

/topology and parameters/species/shake pair

([Integer 1, integer 2, real number 1] × [number of constraints]) "Integer 1" and "integer 2" must be the intramolecular numbers given to the atoms in a constraint. "Real number 1" must be the distance of the constraints. Write this setting as many times as the number of constraints.

<Unit> Integer 1: none, integer 2: none, real number 1: Å

/topology and parameters/species/nbond

(integer×1) "Integer" must be the number of intramolecular bonds (for which a force field is set).

/topology and parameters/species/bond

([Integer 1, integer 2, real number 1, real number 2] × [number of bonds])

"Integer 1" and "integer 2" must be the intramolecular numbers given to the atoms that

form a bond.

"Real number 1" must be the value of k_b.

"Real number 2" must be the ideal bond length.

Write the above setting as many times as the number of bonds.

<Unit> Integer 1: none, integer 2: none, real number 1: (kcal/mol)/(Å^2), real number 2: Å

/topology and parameters/species/nangle

(Integer×1) "Integer" must be the intramolecular angles (for which a force field is set).

/topology and parameters/species/angle

([Integer 1, integer 2, integer 3, real number 1, real number 2] × [number of angles]) "Integer 1," "integer 2," and "integer 3" must be the intramolecular numbers given to the atoms that form an angle. The atoms must be bound to each other in the form of "1-2-3". "Real number 1" must be the value of K_θ. "Real number 2" must be the value of θ_0. Write the above setting as many times as the number of angles.

<Unit> Integer 1, integer 2, and integer 3: none, real number 1: (kcal/mol)/(radian^2), real number 2: degree

/topology and parameters/species/nub

(Integer×1) "Integer" must be the number of ub's in a molecule.

/topology and parameters/species/ub

([Integer 1, integer 2, integer 3, real number 1, real number 2] × [number of ub's]) "Integer 1," "integer 2," and "integer 3" must be the intramolecular numbers given to the atoms that form a ub. The atoms must be bound to each other in the form of "1-2-3". "Real number 1" must be the value of K_{UB}. "Real number 2" must be the value of s_0. Write the above setting as many times as the number of ub's.

<Unit> Integer 1: none, integer 2: none, integer 3: none, real number 1: (kcal/mol)/(Å^2), real number 2: Å

/topology and parameters/species/ndihedral

(Integer×1) "Integer" must be the number of dihedrals in a molecule.

/topology and parameters/species/dihedral

([Integer 1, integer 2, integer 3, integer 4, real number 1, integer 5, real number 2] × [number of dihedrals]) "Integer 1," "integer 2," "integer 3," and "integer 4" must be the intramolecular numbers given to the atoms that form a dihedral. The atoms must be bound to each other in the form of "1-2-3-4". "Real number 1" must be the value of K_X. "Integer 5" must be the value of n. "Real number 2" must be the value of δ. Write the above setting as many times as the number of dihedrals.

<Unit> Integer 1, integer 2, integer 3, and integer 4: none

Real number 1: kcal/mol, integer 5: none, real number 2: degree

/topology and parameters/species/ncmap

(Integer×1) "Integer" must be the number of cmaps in a molecule.

/topology and parameters/species/cmap

([Integer 1, integer 2, integer 3, integer 4, integer 5, integer 6] × [number of cmaps])

"Integer 1" must be the intramolecular number given to an atom of C.

"Integer 2" must be the intramolecular number given to an atom of N.

"Integer 3" must be the intramolecular number given to an atom of CA.

"Integer 4" must be the intramolecular number given to an atom of C.

"Integer 5" must be the intramolecular number given to an atom of N.

"Integer 6" must be the number given to the type of CMAP.

Write the above setting as many times as the number of cmaps.

/topology and parameters/species/nitorsion/nitorsion

(Integer ×1) "Integer" must be the number of improper torsions.

/topology and parameters/species/itorsion

([Integer 1, integer 2, integer 3, integer 4, real number 1, real number 2] × [number of improper torsions]) "Integer 1," "integer 2," "integer 3," and "integer 4" must be the intramolecular numbers given to the atoms that form an improper torsion, where the atom indicated by "integer 1" is the central atom, and the improper torsional angle is defined by the angle between plane "1-2-3" and plane "2-3-4". "Real number 1" must be the value of K_{ψ}. "Real number 2" must be the value of ψ_0. Write the above setting as many times as the number of improper torsions.

<Unit> Integer 1, integer 2, integer 3, and integer 4: none, real number 1: (kcal/mol)/(radian^2), real number 2: degree

9.3 Initial coordinates and velocities (.mdxyz)

/atom/natom

(Integer × 1) Total number of atoms in the system

/atom/positions

([Real number] × [3 × number of atoms]) Write the coordinates of an atom in the order of x, y, and z coordinates, and then write the setting as many times as the number of atoms.
<Unit> Å

/atom/velocities

([Real number] × [3 × number of atoms]) Write the velocities of an atom in the order of x, y, and z directions, and then write the setting as many times as the number of atoms.
<Unit> Å/sec

/thermostat/nthermostat

(Integer × 1) Number of chains of Nosé-Hoover chain thermostats for an isothermal ensemble (nvt or npt_a). The default is 5.

/thermostat/positions

([Real number] × [1 × number of thermostats]) Generalized coordinates of thermostat for particles. Write this setting as many times as the number of thermostats. Typically, the initial value will be "0.0".

/thermostat/velocities

([Real number] × [1 × number of thermostats]) Generalized velocities of thermostat for particles. Write this setting as many times as the number of thermostats. Typically, the initial value will be "0.0".

/barostat/nbarostat

(Integer × 1) Number of chains of thermostats for barostat temperature control for an isobaric ensemble (npt_a). The default is 5.

/barostat/positions

([Real number] × [1 × number of barostats]) Generalized coordinates of thermostat for barostat. Write this setting as many times as the number of chains of thermostats. Typically, the initial value will be "0.0".

/barostat/velocities

([Real number] × [1 × number of barostats]) Generalized velocities of thermostat for barostat. Write this setting as many times as the number of chains of thermostats. Typically, the initial value will be "0.0".

/periodic cell/length

(Real number × 3) Cell lengths in the x, y, and z directions under a periodic boundary condition (This MODYLAS version supports this setting only when the lengths in the x, y, and z directions are the same [that is, when the cell is a cubic cell].)
<Unit> Å

/periodic cell/angle/

(Real number × 3) Angles α, ß, and γ of a rhombic cell. In the case of a rectangular parallelepiped cell, the angles are 90.0 degrees.

<Unit> degree

/periodic cell/vboxg

([Real number] × [3 × 3]) Free velocity of a rhombic cell. Because the setting is a 3 × 3 matrix, list nine real numbers. The initial values may be all "0.0".

9.4 Position constraint of atoms (.posiconst)

In this input file, tags are not used but the following format is used:

First line: Keywords

"no-position-constrain" … Position constraint is not imposed.

"fix," "huge_mass," "harmonic" … See "/position constrain/type" keyword in the mdff file.

Note that, case for using keyword "harmonic," you must also write the value of the force constant [kcal/mol/Å^2] on the first line.

(Example) harmonic 1.0

Second and subsequent lines (same number of lines as the total number of atoms): Write "0" or "1" on each line.

"0" … Atom to be constrained

"1" … Atom not to be constrained

Note that, only case for using keyword "harmonic," you must write the values of the x, y, and z constrained coordinates of constrained atoms after the "0" or "1" on each line.

(Example)

0 1.00 -12.123 2.234

0 0.10 2.245 7.999

1 0.30 1.345 9.999

0 0.20 2.345 2.999

1 23.00 4.01 3.000

… (Same number of lines as the total number of atoms)

10. Reference Documents

[1] "MODYLAS: A Highly Parallelized General-Purpose Molecular Dynamics Simulation Program for Large-Scale Systems with Long-Range Forces Calculated by Fast Multipole Method (FMM) and Highly Scalable Fine-Grained New Parallel Processing Algorithms", *J. Chem. Theo. Comp.*,**9**, 3201-3209 (2013), Yoshimichi Andoh, Noriyuki Yoshii, Kazushi Fujimoto, Keisuke Mizutani, Hidekazu Kojima, Atsushi Yamada, Susumu Okazaki, Kazutomo Kawaguchi, Hidemi Nagao, Kensuke Iwahashi, Fumiyasu Mizutani, Kazuo Minami, Shin-ichi Ichikawa, Hidemi Komatsu, Shigeru Ishizuki, Yasuhiro Takeda, and Masao Fukushima

11. Developers

- Yoshimichi Andoh (Nagoya University)
- Noriyuki Yoshii (Nagoya University)
- Kazushi Fujimoto (Ritsumeikan University)
- Hidekazu Kojima (Nagoya University)
- Atsushi Yamada (Nagoya University)
- Susumu Okazaki (Nagoya University)
- Kensuke Iwahashi (Institute for Molecular Science)
- Fumiyasu Mizutani (Institute for Molecular Science)