omegagene Documentation

Release 0.52

myPresto/omegagene team

Sep 11, 2020

CONTENTS

CHAPTER

ONE

ABOUT MYPRESTO/OMEGAGENE

"myPresto/omegagene" is the molecular dynamics (MD) simulation software. myPresto/omegagene has several unique features, such as, the zero-multipole summation method and the virtual-system coupled sampler.

The current version of omegagene can perform:

1. MD simulations on NVE ensemble
2. MD simulations on NVT ensemble (The velocity rescaling, or Hoover-Evans thermostat)
3. MD simulations on Virtual-system coupled samplers
4. Applying constraints with SHAKE algorithm
5. Calculations of electrostatic potentials based on the zero-dipole summation method.
6. Calculations of pairwise potentials on GPGPU (powered by CUDA 7.0, Computer Capability 3.5 is required).
7. Coarse-grained MD simulations with the hydrophobicity scale model and Debye-Huckel approximation.

CHAPTER

TWO

DEVELOPERS

- KASAHARA Kota, Ritsumeikan Univ., Japan
- TERAZAWA Hiroki, Ritsumeikan Univ., Japan
- ITAYA Hayato, Ritsumeikan Univ., Japan
- GOTO Satoshi, Ritsumeikan Univ., Japan
- TAKAHASHI Takuya, Ritsumeikan Univ., Japan
- MA Benson, Univ. Illinois, US
- GOTO Kota, Tokyo Tech, Japan
- BHASKAR Dasgupta, Osaka Univ., Japan
- HIGO Junichi, Univ. Hyogo, Japan
- FUKUDA Ikuo, Osaka Univ., Japan
- MASHIMO Tadaaki, AIST, Japan
- AKIYAMA Yutaka, Tokyo Tech, Japan
- NAKAMURA Haruki, Osaka Univ., Japan

CHAPTER

THREE

USERS' MANUAL

3.1 System Preparation

Author Kota Kasahara

3.1.1 Input files

The input files required for MD simulations by *omegagene* are compatible with myPresto/psygene. *omegagene* requires

1. Restart file (including the initial coordinates and velocities)
2. Topology file (generated by Tplgene program)
3. Shake file (generated by SHAKEinp program)

In addition, two configuration files are required.

4. System configuration file (system.cfg)
5. Simulation configuration file (md.cfg)

omega_toolkit generates a binary file by combining some of the input files.

```
python2.7 ${OMEGATK}/mdinput_generator.py -i system.cfg -o system.cls -v v.0.52
```

Example of system.cfg file is shown below:

system.cfg:

```
--fn-i-tpl          et1.tpl     ; Topology file
--fn-i-initial-pdb  et1.pdb     ; Structure file in .pdb format
--fn-i-restart      et1.restart ; Restart file
--cell-x            61.2425     ; Specifying the each length of cell-axes
--cell-y            61.2425
--cell-z            61.2425
--fn-i-shake        system.shk  ; SHAKE file
;--fn-i-ttp-v-mcmd-inp      ttp_v_mcmd.inp  ; For V-McMD
;--fn-i-ttp-v-mcmd-initial  start.vert      ; For V-McMD
```

md_input.cfg:

```
--mode                  md                  ; Only the keyword "md" is accepted
--integrator            leapfrog-presto     ; "leapfrog-presto" or "zhang"
--thermostat            scaling     ; Thermostat algorithm
                                    ; "none" or "scaling"
--cutoff                12.0        ; Cutoff distance in angestrome
--n-steps               10          ; The number of steps to be calculated
--time-step             2.0         ; Integration time step in fs
--electrostatic         zero-dipole ; Only "zero-dipole" is accepted for GPU
                                    ; "zero-quadrupole", "zero-octupole", and
```

```
                                    ; "zero-hexadecapole" are also aceppted for CPU.
--ele-alpha             0           ; Dumping factor for ZD
                                    ;   only 0 is OK for GPU version
--temperature           300         ; The target temperature
;--temperature-init     10          ; The initial temperature
;--heating-steps        10000       ; The number of steps for heating
--print-interval-log    1           ; Interval steps for printing logs
--print-interval-coord  1           ; Interval steps for output the trajectory
--fn-o-coord            trpc.trr    ; File name for the output trajectory
--format-o-coord        presto      ; File format for the output trajectory
                                    ; only "presto" is allowed.
;--fn-o-log              et1.log     ; Not used in the current version
;--fn-o-energy           et1.ene     ; Not used in the current version
--nsgrid-cutoff         13.0        ; Neighbor search cutoff radius (angstrom)
--nsgrid-update-intvl   50          ; Interval time steps for update the neighbor search grid

; Expanded ensemble
;--expanded-ensemble             v-mcmd          ; "none", "v-mcmd", or "v-aus"
;--fn-o-vmcmd-log                ttp_v_mcmd.out ; Output filename
;--fn-o-expand-lambda            mule.ene        ; Output filename
;--print-interval-expand-lambda  1               ; Interval steps for the output
;--format-o-expand-lambda        ascii           ; "ascii", or "binary"
```

3.1.2 Execute

omegagene executes a MD simulation with two configuration files provided as command line arguments. The current version of omegagene output log messages to the standard output. Redirecting to a log file is recommended.

```
omegagene --inp et1.cls --cfg md_input.cfg > log.txt
```

3.2 In/Out Files

3.2.1 Input Files

1. System configuration file (.cfg)
2. Simulation configuration file (.cfg)
3. Structure file (.pdb)
4. Initial coordinates and velocities file (.restart)
5. Topology file (.tpl)
6. Integrated binary (.cls)
7. SHAKE setting file (.shk)
8. V-McMD (or V-AUS) setting files (.inp, .vert)
9. V-AUS restart file (.dat)
10. Atom group definition file (.inp)
11. Distance restraint file (.inp)
12. Position restraint file (.inp)
13. VcMD parameter file (.inp)
14. VcMD initial state file (.inp)

System configuration file (.cfg)

The input file describing configurations about a simulation system. sOther some input files are specified in this file, and they are integrated into the integrated binary file (.cls) by using *mdinput_generator.py* program. In the configuration file, a set of a key and value(s) is specified in each line.

- **--fn-i-tpl** md.tpl
  - The file name of the topology file (.tpl).
- **--fn-i-initial-pdb** md.pdb
  - The file name of the structure file (.pdb).
- **--fn-i-restart** md.restart
  - The file name of the initial coordinates and velocities file (.restart)
- **--cell-x** 61.2425
- **--cell-y** 61.2425
- **--cell-z** 61.2425
  - The lengths of the periodic boundary cell in each axis in angestrome unit.
- **--fn-i-shake** system.shk
  - The file name of the shake setting file.
- **--fn-i-ttp-v-mcmd-inp** ttp_v_mcmd.inp
- **--fn-i-ttp-v-mcmd-initial** start.vert
  - The file name of the V-McMD setting files.
- **--fn-i-atom-group** atom_groups.inp
  - The file name of the atom group definition file.
- **--fn-i-dist-restraint** dist_rest.inp
  - The file name of the distance restraint setting file.
- **--fn-i-aus-restart** aus_rest.dat

Simulation configuration file (.cfg)

The input file describing configurations about simulation conditions.

Common configurations

- **--mode** md
  - Only the keyword “md” is valid.
- **--gpu-device-id** 0
  - The device ID of GPGPU board to be used.
- **--integrator** leapfrog-presto
  - Type of integrator
  - leapfrog-presto * The leap frog algorithm. * For NVE, NVT (rescaling), SHAKE
  - zhang * The integrator by Zhang [Zhang_1997] * For the Hoover-Evans thermostat (“hoover-evans”) * SHAKE cannot be applied.
  - langevin * Langevin integrator. * The thermostat must be set to “none” * “langevin-gamma” is required.

- **--thermostat** scaling
  - none * NVE
  - scaling * Scaling velocities.
  - hoover-evans * Hoover-Evans. This can be applied for the integrator "zhang".
- **--cutoff** 12.0
  - The cutoff length for non-bonded potential (angestrome unit)
- **--n-steps** 10
  - The number of steps of the simulation.
- **--time-step** 2.0
  - The integration time step
- **--electrostatic** zero-dipole
  - "zero-dipole" The Zero-dipole summation (ZD) method developed by Fukuda [Fukuda_2011] .
  - "zero-quadrupole"
  - "zero-octupole"
  - "zero-hexadecapole" The Zero-multipole summation (ZM) method developed by Fukuda [Fukuda_2014] .
  - "debye-huckel" For coarse-grained simulation. For GPU mode, only zero-dipole with –ele-alpha 0.0 is acceptable. To use "debye-huckel" with the HPS model with GPU, omegagene must be built with -DCELESTE_HPSGPU option.
- **--ele-alpha** 0.0
  - The dumping factor fo ZM method.
  - For GPU mode, only 0.0 is acceptable
- –debye-huckel-dielectric * A parameter for Debye-Huckel approx.
- –debye-huckel-ionic-strength * A parameter for Debye-Huckel approx.
- **--debye-huckel-temperature** 300
  - A parameter for Debye-Huckel approx.
- **--nonbond** lennard-jones
  - "lennard-jones" * Usual LJ potential.
  - "hydrophobicity-scale-lj" LJ potential modified with the hydrophobicity scale model. To use this otpion on a GPU, omegagene must be built with -DCELESTE_HPSGPU=1 option for cmake.
- **--temperature** 300
  - Temperature for the thermostat.
- **--temperature-init** 300
  - The initial temperature. Default value is the temperature specified by "–temperature" setting.
  - With this setting, "–heating-steps" should be set.
- **--heating-steps** 0
  - The temperature is linearly increased or decreased from the –temperature-init to –temperature during the steps specified in this setting.
- **--berendsen-tau** 0.2
  - Tau parameter for the Berendsen thermostat.
- **--langevin-gamma** 0.1

  - Gamma parameter for the Langevin integrator.
- **--print-interval-log** 1
  - Output interval for the log (the standard output)
- **--print-interval-coord** 1
  - Output interval for the trajectory file.
- **--fn-o-coord** et1.trr
  - Output file name for the trajectory file.
- **--format-o-coord** presto
  - The file format of the trajectory.
  - presto
- **--fn-o-restart** md.restart
  - Output restart file name.
- **--nsgrid-cutoff** 13.0
  - The cutoff length for the neighbor search (angestrome unit).
- **--nsgrid-update-intvl** 10
  - The interval steps for execution of the neighbor search.
- **--com-motion** cancel
  - Settings for canceling the center of mass
  - none
  - cancel * Translation of the center of mass for some specified groups are cancelled. * pThe groups should be specified in "–com-cancel-group-name"
- **--com-cancel-group-name** grpA
  - The name of an atom group COM motions of which to be cancelled.
  - Multiple values can be specified.

Configuration for restraints

- –dist-restraint harmonic * Functions for distance restraints * none * harmonic
- –dist-restraint-weight * Scaling coefficient for the distance restraints
- –position-restraint * none * harmonic
- –position-restraint-weight * Scaling coefficient for the

Configuration for the extended ensemble methods

- **--extended-ensemble** v-mcmd
  - none Extended ensemble is not used
  - v-mcmd The V-McMD method [Higo et al. (2013)]_
  - v-aus The V-AUS method [Higo et al. (2015)]_
  - vcmd The VcMD emthod [Higo et al. (2017a)]_ [Higo et al. (2017b)]_ [Hayami et al. (2018)]_ [Hayami et al. (2019)]_
- **--fn-o-vmcmd-log** ttp_v_mcmd.out

  - Output file name of a virtual-system trajecotry.
- **--fn-o-extended-lambda** mule.ene
  - Output file name of a log of the lambda value.
- **--print-interval-extended-lambda** 1
  - Output interval for the log of the lambda value.
- **--format-o-extended-lambda** ascii
  - File format of the log of the lambda value.
  - ascii
  - binary
- –enhance-sigma * A parameter for the recovery force in V-AUS simulation. * A margin of the lambda value.
- –enhance-recovery-coef * Strength of the recovery force which works when the reaction coordinate is out of the predefined range.
- –fn-o-vcmd-start * Output file name describing the virtual state in the last step. This can be used for restarting the VcMD runs.
- –fn-o-vcmd-q-raw * Output file name for the populations of each virtual state.
- –fn-o-vcmd-q-raw_is * Output file name for the populations of each intersection of virtual states.
- –begin-count-q-raw * The step number which begans to count the populations.
- –vcmd-drift * 0 or 1. * 1 indicates the simulations with drifting in the virtual system without detailed-balance.
- –print-interval-group-com * Interval steps for output the reaction coordinate values.

Initial coordinates and velocityies file (.restart)

This file is compatible for myPresto/Psygene restart file. For the first run, this file with random velocities can be generated by *presto_generate_velocities.py* scripts in the toolkit. At the end of a simulation, final coordinates and velocities will be output at the file specified by *–fn-o-restart* option.

Topology file (.tpl)

This file is compatible for myPresto/Psygene topology file. It can be prepared by using myPresto/TopolgeneX program.

- –fn-i-tpl md.tpl

SHAKE setting file.

This file is compatible for myPresto/Psygene SHAKE file. It can be prepared by using SHAKEinp program.

It should be specified in the system configuration file:

- –fn-i-shake system.shk

V-McMD (or V-AUS) setting files (.inp, .vert)

These files are compatible for myPresto/Psygene files.

They should be specified in the system configuration file:

- **--fn-i-ttp-v-mcmd-inp** ttp_v_mcmd.inp
- **--fn-i-ttp-v-mcmd-initial** start.vert

ttp_v_mcmd.inp

This file describes the definition of virtual states and their bias functions.

```
;
3                       # The number of virtual states (VSs).
100                     # Interval steps for VS transitions.
;
-37700.0   -35560.0     # The lower and upper bound of the 1st VS
0.0  1.0                # Transition probability for lower and upper VS
;
-36630.0   -34490.0     # The 2nd VS.
1.0  1.0                #
;
-35560.0   -33420.0     # The 3rd VS.
1.0  1.0                #
;
6                       # The bias function of 1st VS is 6-th order polynomial
0.539566614289679E+06   # The parameters of the polynomial
0.107078199910006E+03
0.885349382820406E-02
0.390387519416188E-06
0.968208493740114E-11
0.128058948306820E-15
0.705682546575764E-21
-0.225508543444448E-01  # The last two values are not used
0.237465821192018E-01   # The last two valuse are not used
6                       # The 2nd VS.
0.153644514042787E+06
0.313777316547802E+02
0.266888476716589E-02
0.121017618153941E-06
0.308531417224255E-11
0.419329550527650E-16
0.237358207968186E-21
-0.502410851186141E-01
-0.443300654878840E-01
6                       # The 2nd VS.
-0.607303057237411E+06
-0.129555612396211E+03
-0.115143655222501E-01
-0.545716541236728E-06
-0.145465976138503E-10
-0.206775621289763E-15
-0.122453436139314E-20
0.201226906548982E+00
-0.190884220501175E-03
300                     # Temperature
```

start_vert.inp

This file describes the initial virtual state and a random seed.:: 2 # The initial virtual state 94265278 # The random seed

V-AUS restart file (.dat)

This file is required for continuing a finished V-AUS job. It should be specified in the system configuration file:

- –fn-i-aus-restart aus_rest.dat

At the end of a V-AUS job, this file is automatically generated at the file, specified by *–fn-o-aus-restart* option.

Atom group definition file (.inp)

This file defining groups of atoms. This informations are used for:

- Canceling the center of mass motion
- Defining enhaced groups for V-AUS simulations

In this ascii file, each line defines one atom group. The characters at the head of each line indicate the name of each group. Successive columns specify atoms in this group.

For example:

```
group1 1 4 6-9
group2 3 10-11 13
```

The group1 is composed of atoms 1, 4, 6, 7, 8, and 9. The group2 is composed of atoms 3, 10, 11, and 13.

The atom-ID are began from 1.

Distance restraint file (.inp)

This file defining the distance restraints between pairs of atoms. Each line between the keywords "RDDSTC> LIST" and "RDDSTC> STOP" defines a restraint. Distances between two atoms were restrained with the flat bottom potential. Each line specifies identities of two atoms, restraint forces for lower and upper bound, and lower and upller borders to apply restraint potential.

- Molecule-ID of the 1st atom
- Residue-ID of the 1st atom
- Residue name of the 1st atom
- Atom name of the 1st atom
- Molecule-ID of the 2nd atom
- Residue-ID of the 2nd atom
- Residue name of the 2nd atom
- Atom name of the 2nd atom
- The force coefficient for the lower bound.
- The force coefficient for the upper bound.
- Lower limit of the distance.
- Upper limit of the distance.

Position restraint file (.inp)

Each line specifies a position restraint for an atom.

- Atom-ID
- Equilibrium X coordinate

- Equilibrium Y coordinate
- Equilibrium Z coordinate
- Margin distance from the equilibrium position.
- Force coefficient.
- Type of restraint. “normal” or “z”. Restraint with “z” omits the X and Y coordinates.

VcMD parameter file (.inp)

This file describes the definition of the virtual system and potentials for each states.

:: 100 ; The interval steps for virtual state transitions. 2 ; The number of dimensions 3 group1 group2 ; The number of virtual states, and names of two groups

; defining the reaction coordinate, for the 1st axis.

3.0 5.0 ; The range of the reaction coordinates for the 1st state. 4.0 6.0 ; That for the 2nd state. 5.0 7.0 ; That for the 3rd state. 4 group1 group3 ; The definition for the second axis. 3.0 5.0 4.0 6.0 5.0 7.0 6.0 8.0 END

VcMD initial state file (.inp)

This file specifies the initial virtual state.:: 2 ; The number of dimension 5 ; Initial virtual state coordinate of the 1st dimension 6 ; Initial virtual state coordinate of the 2nd dimension 46582642 ; Random seed

3.2.2 Output files

1. Standard output
2. Trajectory file (.cod)
3. Restart file (.restart)
4. V-McMD (or V-AUS) lambda trajectory
5. V-McMD (or V-AUS) virtual-system trajectory
6. V-AUS restart file

A simulation log will be output for the standard output. Redirection to a file is recommended.

The trajectory file format is compatible to myPresto/Psygene.

This file repeats the two pars: a header of the frame, and atomic coordinates at the frame.

For the header part:

- [4 bytes, INT] The size of header parts in bytes. Always “44”.
- [4 bytes, INT] Step number
- [4 bytes, FLOAT] Time
- [4 bytes, FLOAT] CPU time
- [4 bytes, FLOAT] Total energy
- [4 bytes, FLOAT] Kinetic energy
- [4 bytes, FLOAT] Temperature
- [4 bytes, FLOAT] Potential energy
- [4 bytes, FLOAT] Always “0.0”
- [4 bytes, FLOAT] Always “0.0”

- [4 bytes, FLOAT] Always “0.0”
- [4 bytes, FLOAT] Always “0.0”
- [4 bytes, INT] The size of header parts in bytes. Always “44”.

For the coordinates part:

- [4 bytes, INT] The size of this part. The nubmer of atom * 3 dimensions * 4 bytes.
- [FLOAT] X, Y, and Z coordinates of each atoms.
- [4 bytes, INT] The size of this part. The nubmer of atom * 3 dimensions * 4 bytes.

The restart file format is compatible to myPresto/Psygene.

This file is composed of the three pars: a header of the frame, atomic coordinates, and velocities.

For the header part:

- [4 bytes, INT] The length of the following text. Alwasy “80”.
- [80 bytes, CHAR] Description of this simulation (version information)
- [4 bytes, INT] Alwasy “80”.
- [4 bytes, INT] Alwasy “8”.
- [4 bytes, INT] The number of atoms for coordinates.
- [4 bytes, INT] The number of atoms for velocities.
- [4 bytes, INT] Alwasy “8”.
- [4 bytes, INT] Alwasy “36”.
- [4 bytes, INT] Step number.
- [4 bytes, FLOAT] Time.
- [4 bytes, FLOAT] Total energy.
- [4 bytes, FLOAT] Kinetic energy.
- [4 bytes, FLOAT] Potential energy.
- [4 bytes, INT] Alwasy “36”.

For the coordinates part:

- [4 bytes, INT] The size of this part in bytes. The number of atoms * 3 dimenstions * 8 bytes.
- [DOUBLE] X, Y, Z coordinates of each atom.
- [4 bytes, INT] The size of this part in bytes. The number of atoms * 3 dimenstions * 8 bytes.

For the velocity part:

- [4 bytes, INT] The size of this part in bytes. The number of atoms * 3 dimenstions * 8 bytes.
- [DOUBLE] X, Y, Z velocities of each atom.
- [4 bytes, INT] The size of this part in bytes. The number of atoms * 3 dimenstions * 8 bytes.

For V-McMD or V-AUS simulations, the lambda values are written on this file.

When *–format-o-extended-lambda ascii* is specified, a lambda value is recorded in each line of the ascii file.

When *–format-o-extended-lambda binary*, is specified, the values are dumped as a binary file.

- [1-4 bytes] The magic number
- [5-8 bytes] The precision (4 or 8)
- [9-14 bytes] Always 1.
- [After that] The lambda values

The trajectory of virtual-system coordinates is written as a two-columns, tab separated table.

- The first column means the step number.
- The second column means the virtual-system coordinate.

For example, in the case that virtual-system transitions are done in every 1000 steps,:

```
1     1
1001  2
3001  1
4001  2
5001  3
```

A binary file required for restarting V-AUS and VcMD simulations.

3.3 Analysis

Author Kota Kasahara

3.3.1 Log

Calculation log is printed to the standard output. The total energy of the system should be constant, in the microcanonical ensemble. When the total energy drifts, you should take care of the following points:

1. Increase –nsgrid-cutoff
2. Decrease –nsgrid-update-intvl
3. Increase –cutoff

```
Step:        0    Time:     0.0000
Total:     -7.4102976562e+04
Potential: -8.7917343750e+04    Kinetic:  1.3814368164e+04
Bond:      2.2974748345e+02    Angle:    9.5691961023e+01
Torsion:   2.9869403994e+02    Improper: 2.9828235618e+00
14-VDW:    1.0501907707e+02    14-Ele:   1.8181411150e+03
VDW:       1.5527498751e+04    Ele:      -1.0599511706e+05

Step:   100000    Time: 50000.0000
Total:     -7.4026648438e+04
Potential: -8.7759882812e+04    Kinetic:  1.3733237305e+04
Bond:      9.4730405123e+03    Angle:    2.4954034353e+02
Torsion:   3.2906814547e+02    Improper: 1.2255384929e+01
14-VDW:    1.2528642296e+02    14-Ele:   1.8064233115e+03
VDW:       1.7495527442e+04    Ele:      -1.1725102074e+05
```

3.3.2 Trajectory

omegagene output the trajectory in the format compatible with myPresto/psygene-G. If you want to convert the trajectory file into Gromacs .trr format, the script to do it is included in *omega-toolkit*.

python ${OMEGATK}/convert_trajectory_presto.py –i-pdb initia.pdb –i-crd traj.crd -o traj.trr

3.4 Samples

samples directory includes some sample simulation data.

- ala3

- cg_q8
- cg_q8_vcmd
- mcmd_ala3
- trpc

3.4.1 ala3

A sample for explicitly-solvated all-atom simulation. This simulation model includes three molecules of capped-Ala peptide. The simulation can be carried out by executing *run.bash* script. Output files for a short simulation (100 steps) with the NVE ensemble also attached. Users can test the built binary by using this data in terms of consistency with the attached output. The potential energies in each step are recorded in *log03_md.txt* file.

3.4.2 cg_q8

A sample data for coarse-grained simulations. See the "Tutorial for Coarse-grained simulations" in this documentation.

3.4.3 cg_q8_vcmd

A sample data for VcMD simulation with the coarse-grained model. See the "Tutorial for Coarse-grained simulations" in this documentation.

3.4.4 mcmd_ala3

A sample data for the McMD simulation. See the "Tutorial for multi-canonical MD (McMD) simulations" in this documentation.

3.4.5 trpc

A sample data for all-atom simulation. This was used for older versions.

CHAPTER

FOUR

BUILD MANUAL

4.1 Installation

Contents

omegagene is written in C++, and its build system utilizes CMake. The following is a non-exhaustive description for building omegagene. Currently there are three compile options of omegagene:

1. CPU (without the compile option)
2. CPU without the neighbor search algorithm (-DCELESTE_WO_NS=1)
3. CPU with GPU (CUDA) (-DCELESTE_GPU=1)
4. CPU with GPU (CUDA) for coarse-grained simulations (-DCELESTE_GPUHPS=1)

For platform-specific details on building omegagene, please refer to the *Celeste Build Notes*.

option	GPU	Neighbor-search	all-atom	coarse-grained
none	NA	enable	enable	NA
-DCELESTE_WO_NS	NA	NA	enable	enable
-DCELESTE_GPU	enable	enable	enable	NA
-DCELESTE_GPUHPS	enable	enable	NA	enable

4.1.1 Software Requirements

- CMake 3.4+
- For the GPU version of omegagene, CUDA 7.0+ is required for C++11 support.
- **A C++11 compiler:**
  - GCC: 4.8+
  - Clang: 3.6+
  - AppleClang: 5.0+

  - Intel: 15.0+ (minimal version required by CUDA 7.0+)
- OpenMP 3.1+
- Python 2.7.x
- numpy

4.1.2 Building omegagene - Standard Version

1. Set up a target build folder:

```
# in <PROJECT_ROOT> directory
localhost$ mkdir target
localhost$ cd target
```

2. Configure the build. CMake will determine all the external software dependencies for the selected build variant, and exit with errors if the dependency requirements are not met. CMake must be invoked on the `CMakeLists.txt` file in the <PROJECT_ROOT> directory:

```
# in <PROJECT_ROOT>/target directory
localhost:target local$ cmake ..
```

3. Build the software:

```
# The verbose flag is optional
localhost:target local$ make VERBOSE=1
```

4.1.3 Building omegagene Without Neighbor Search Routines

While neighbor search is effective for fast calculations, the implementation is complicated and may be difficult to debug MD runs. For this reason, a version of omegagene without the neighbor search routines can be built for debugging or testing.

To build this version of omegagene, simply run the following command instead when configuring the build (Step 2):

```
localhost:target local$ cmake -DCELESTE_WO_NS=1 ..
```

The compiled executable will be named `omegagene_wons`.

4.1.4 Building omegagene With GPU Acceleration

For building this version of omegagene, CUDA 7.0+ is required. For running the binary, an NVIDIA GPU with Compute Capability >= 3.5 or later is required.

To build this version of omegagene, simply run the following command instead when configuring the build (Step 2):

```
localhost:target local$ cmake -DCELESTE_GPU=1 ..
```

CMake will automatically determine the default installation paths for the CUDA libraries and `nvcc`. Please refer to the Build Notes if you have installed CUDA to a custom filesystem path.

The compiled executable will be named `omegagene_gpu`.

4.1.5 Building omegagene for a coarse-grained method

For the coase-grained simulation on GPU, *-DCELESTE_GPUHPS=1* option is required.

```
localhost:target local$ cmake -DCELESTE_GPUHPS=1 ..
```

For CPU, the coarse-grained simulation can be performed by the binary with *-DCELESTE_WO_NS=1* option.

4.1.6 omegagene Toolkit

omegagene tookit is a library of pre- and post-processing scripts for MD simulations to be used with omegagene It requires Python 2.7.x and the `numpy` library.

This manual assumes that the omegagene toolkit directory specified in the environmental variable `${OMEGATK}`. This path should be added in `${PYTHONPATH}`:

```
export OMEGATK="${HOME}/omegagene/toolkit"
export PYTHONPATH=${OMEGATK}:${PYTHONPATH}
```

```
setenv OMEGATK "${HOME}/omegagene/toolkit"
setenv PYTHONPATH ${OMEGATK}:${PYTHONPATH}
```

4.2 Celeste Build Notes

Author Benson Ma

Contents

Below is an assortment of build notes for handling different software dependencies and different platforms.

4.2.1 General

CUDA

- Library code that is built on top of CUDA must be built as **shared libraries**; otherwise, linker errors will appear during building on Linux platforms. Hence, the entries in `CMakeLists.txt` specifying building CUDA-dependent libraries should be marked `SHARED` as such:

  ```
  CUDA_ADD_LIBRARY(CelesteFooCUDA SHARED foo.cu bar.cu)
  ```

- The version of gcc installed may be a later version than the the latest officially-supported host compiler for `nvcc`. You will see an error like this:

```
# in <PROJECT_ROOT>/target
localhost:target local$ cmake -D CMAKE_C_COMPILER=/opt/local/bin/gcc-mp-5 -D␣
↪CMAKE_CXX_COMPILER=/opt/local/bin/g++-mp-5 -D CELESTE_GPU=1 ..
...
... generating Makefiles
...

localhost:target local$ make
...
... building code
...
/usr/local/cuda/include/host_config.h:115:2: error: #error -- unsupported GNU␣
↪version! gcc versions later than 4.9 are not supported!
```

While not recommended, this can be fixed by commenting out the appropriate `#error` macro in `<CUDA_ROOT>/include/host_config.h`:

```
#if __GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ > 9)

// #error -- unsupported GNU version! gcc 4.10 and up are not supported!

#endif /* __GNUC__> 4 || (__GNUC__ == 4 && __GNUC_MINOR__ > 9) */
```

4.2.2 Linux

MPI

- Installation of MPICH or OpenMPI may not include adding `mpicc`/`mpic++` to the `$PATH`, resulting in the following error when `cmake` is invoked:

  ```
  Could NOT find MPI_C (missing: MPI_C_LIBRARIES MPI_C_INCLUDE_PATH)
  ```

  To resolve this, simply add the directory containing `mpicc`/`mpic++` to the `$PATH` in the `ENVIRONMENT` or in the *cmake'* invocation:

  ```
  localhost:target local$ PATH=$PATH:/usr/lib64/mpich/bin cmake ..
  ```

4.2.3 Mac OS X

- Unfortunately, with the newer versions of CUDA on the Mac, gcc is not a supported host compiler. Attempting to compile CUDA code using gcc as the host compiler will result in an error message that looks like this:

  ```
  # in <PROJECT_ROOT>/target
  localhost:target local$ cmake -D CMAKE_C_COMPILER=/opt/local/bin/gcc-mp-5 -D␣
  ↪CMAKE_CXX_COMPILER=/opt/local/bin/g++-mp-5 -D CELESTE_GPU=1 ..
  ...
  ... generating Makefiles
  ...

  localhost:target local$ make
  ...
  ... building code
  ...
  nvcc fatal   : GNU C/C++ compiler is no longer supported as a host compiler on␣
  ↪Mac OS X.
  ```

  Intel's ICC does not appear to be a supported host compiler for CUDA on Mac OS X either. Only Clang appears to be a supported host compiler, but this only applies to "AppleClang" (the

version of Clang maintained by Apple). Attempting to use (newer versions of) mainline Clang will result in an error message that looks like this:

```
# in <PROJECT_ROOT>/target
localhost:target local$ cmake -D CMAKE_C_COMPILER=/opt/local/bin/clang-mp-3.7 -D␣
↪CMAKE_CXX_COMPILER=/opt/local/bin/clang++-mp-3.7 -D CELESTE_GPU=1 ..
...
... generating Makefiles
...

localhost:target local$ make
...
... building code
...
nvcc fatal   : The version ('30700') of the host compiler ('clang') is not␣
↪supported
```

While not recommended for ABI/linking reasons, issues such as this above can be resolved by specifying a _different_ compiler as the host compiler for nvcc:

```
# where /usr/bin/clang symlinks to AppleClang
localhost:target local$ cmake -D CMAKE_C_COMPILER=/opt/local/bin/clang-mp-3.7 -D␣
↪CMAKE_CXX_COMPILER=/opt/local/bin/clang++-mp-3.7 -D CELESTE_GPU=1 -D CUDA_HOST_
↪COMPILER=/usr/bin/clang ..
```

4.2.4 Windows

- MSVC does not define the alternative tokens for logical operators (i.e. and in place of &&) by default. See http://stackoverflow.com/questions/24414124/why-does-vs-not-define-the-alternative-tokens-for-logical-operators. This issue can be circumvented by including the following header in source files that use alternative tokens:

  ```
  #include <ciso646>
  ```

  The correct solution is to disable C++ language extensions in MSVC by use of the /Za compiler flag; however this flag is known to be buggy and will result in ODR errors during linking. See the following articles:

  - http://cidebycide.blogspot.com/2015/10/visual-studio-2015-icu-and-error-lnk2005.html
  - http://stackoverflow.com/questions/31808256/multi-file-iostream-error-lnk2005-in-vs2015-with-za

CHAPTER

FIVE

TUTORIAL

5.1 Tutorial for Coarse-grained simulations

5.1.1 Requirements

The following environments are required to use myPresto/omegagene.

- Linux or macos system
- Python with numpy and scipy libraries
- cmake
- c++ 11 or later

In addition, use of bash environment is assumed in this document. Some trivial changes are needed for csh users.

5.1.2 Download myPresto/omegagene

To download myPresto/omegagene, execute the following command on your terminal:

```
$ git clone https://github.com/kotakasahara/omegagene.git
```

We also offer you to use our docker container.

```
git clone https://github.com/terapizawa/myPresto-omegagene.git
```

5.1.3 Installation

Setting up a target build folder:

```
# in ${PROJECT_ROOT} directory
$ mkdir target
$ cd target
```

The **$ {PROJECT_ROOT}** indicates the path to the directory of the downloaded omegagene repository in your environment. Then, CMake evaluates all the external software dependencies for the selected build variant, and exit with errors if the dependency requirements are not met. CMake must be invoked on the *CMakeLists.txt* file in the **$ {PROJECT_ROOT}** directory. Run the following command to configure for building the desired variant of myPresto/omegagene in **$ {PROJECT_ROOT}/target** directory

```
$ cmake -DCELESTE_WO_NS=1 ..
```

When you use the GPU version of myPreto/omegagene, apply the option *-DCELESTE_GPUHPS=1* as follows:

```
$ cmake -DCELESTE_GPUHPS=1 ..
```

Then, *make* command compiles and builds the software.

```
$ make
```

See also "Installation" and "Build manual" in this documentation for details. The executable binary is generated in ${PROJECT_ROOT}/target/bin directory.

5.1.4 Setting up input Files

1. Set paths

```
$ export OMEGATK=${PROJECT_ROOT}/toolkit
$ export OMEGABIN=${PROJECT_ROOT}/target/bin/omegagene_wons
```

You should change these settings depending on your environment. If you use csh environment, *setenv* command should be used to set the environmental variables.

2. Seting up the directory to the simulation

```
$ mkdir ${PATH_TO_YOUR_WORKING_DIRECTORY}
```

Set the variable ${PATH_TO_YOUR_TARGET_DIRECTORY} as the path to your working directory. (e.g. /home/user/md/test)

Then, copy the sample files to the working directory.

```
$ cp -r ${PROJECT_ROOT}/samples/cg_q8 ${PATH_TO_YOUR_WORKING_DIRECTORY}
$ cd ${PATH_TO_YOUR_WORKING_DIRECTORY}/cg_q8
```

3. make a topology file

This sample system consists of two molecules of poly-Q octapeptides. See *inp.pdb* file. To make the topology file, a .pdb file consisting of single molecule is needed.

By using an editor, copy the atoms in the first molecule in *inp.pdb* (the first eight lines) and paste to a new file, and save it as *single.pdb*.

inp.pdb:

```
ATOM      1  CA  GLN A   1     -20.000 0.00000 -15.200  1.00  0.00      0.0X
ATOM      2  CA  GLN A   2     -20.000 0.00000 -11.400  1.00  0.00      0.0X
ATOM      3  CA  GLN A   3     -20.000 0.00000 -7.6000  1.00  0.00      0.0X
ATOM      4  CA  GLN A   4     -20.000 0.00000 -3.8000  1.00  0.00      0.0X
ATOM      5  CA  GLN A   5     -20.000 0.00000 0.00000  1.00  0.00      0.0X
ATOM      6  CA  GLN A   6     -20.000 0.00000 3.80000  1.00  0.00      0.0X
ATOM      7  CA  GLN A   7     -20.000 0.00000 7.60000  1.00  0.00      0.0X
ATOM      8  CA  GLN A   8     -20.000 0.00000 11.4000  1.00  0.00      0.0X
ATOM      9  CA  GLN B   1     0.00000 0.00000 -15.200  1.00  0.00      0.0X
ATOM     10  CA  GLN B   2     0.00000 0.00000 -11.400  1.00  0.00      0.0X
ATOM     11  CA  GLN B   3     0.00000 0.00000 -7.6000  1.00  0.00      0.0X
ATOM     12  CA  GLN B   4     0.00000 0.00000 -3.8000  1.00  0.00      0.0X
ATOM     13  CA  GLN B   5     0.00000 0.00000 0.00000  1.00  0.00      0.0X
ATOM     14  CA  GLN B   6     0.00000 0.00000 3.80000  1.00  0.00      0.0X
ATOM     15  CA  GLN B   7     0.00000 0.00000 7.60000  1.00  0.00      0.0X
ATOM     16  CA  GLN B   8     0.00000 0.00000 11.4000  1.00  0.00      0.0X
```

single.pdb:

```
ATOM      1  CA  GLN A   1     -20.000 0.00000 -15.200  1.00  0.00      0.0X
ATOM      2  CA  GLN A   2     -20.000 0.00000 -11.400  1.00  0.00      0.0X
ATOM      3  CA  GLN A   3     -20.000 0.00000 -7.6000  1.00  0.00      0.0X
ATOM      4  CA  GLN A   4     -20.000 0.00000 -3.8000  1.00  0.00      0.0X
ATOM      5  CA  GLN A   5     -20.000 0.00000 0.00000  1.00  0.00      0.0X
```

```
ATOM      6  CA  GLN A   6     -20.000 0.00000 3.80000  1.00  0.00      0.0X
ATOM      7  CA  GLN A   7     -20.000 0.00000 7.60000  1.00  0.00      0.0X
ATOM      8  CA  GLN A   8     -20.000 0.00000 11.4000  1.00  0.00      0.0X
```

Then, conduct the following command to generate the topology file, *md.tpl*.

```
$ python2.7 ${OMEGATK}/gen_tpl.py --pdb single.pdb --param param.dat --tpl md.tpl --molname␣
↪mol1
```

After that, change the number of molecules in the md.tpl.

md.tpl,:

```
TPL> TITLE
MD

TPL> MOLECULES
mol1                          2  ## <== HERE. This value must be "2".

TPL> ATOMS
mol1
```

4. Generate the initial atomic velocities

Execute the following command.

```
python2.7 ${OMEGATK}/presto_generate_velocities.py   -i inp.pdb   --i-tpl md.tpl   -t 100   -o␣
↪md.restart   -s ${RANDOM}  --mol --check
```

-s option indicates the random seed.

bash environment automatically generates a random number for the ${RANDOM} variable. When it does not work, replace ${RANDOM} into an arbitral arbitral number.

5. make a cls file

```
python2.7 ${OMEGATK}/mdinput_generator.py -i md.inp -o md.inp.cls -v v.0.52 > log_inputgen.txt
```

md.inp.cls file is an input file for myPresto/omegagene.

5.1.5 Setting up your simulation conditions

The simulation conditions and systems are configured by the following files.

- atom_groups.inp
- md.inp
- md.inp.run

atom_groups.inp

```
mol1 1-8 # amino No for each molecules
mol2 9-16
all 1-16  # all amino acids in the input PDB file
```

md.inp

```
--fn-i-tpl              md.tpl          # tpl file for the simulations
--fn-i-initial-pdb      inp.pdb          # input PDB files
--fn-i-restart          md.restart       # all initial positions for the input PDB file
--cell-x                50              # maximum range of x axis
--cell-y                50              # maximum range of x axis
--cell-z                50              # maximum range of x axis
```

```
--cell-center-x          0              # center position for x axis
--cell-center-y          0              # center position for y axis
--cell-center-z          0              # center position for z axis
--fn-i-atom-groups       atom_groups.inp  # information for all amino acids and its molecules
```

md.inp.run

```
--processor                   single        ;    # the numner of processors for conducting MD
--gpu-device-id               0                  # GPU device ID for conducting MD
--mode                        md            ;    # simulation mode
--integrator                  langevin  ;        # the method of integration
--langevin-gamma              1.0   ;      ;     # the parameter for friction coefficient
--cutoff                      20.0          ;    # the cut-off distance in angstrome
--n-steps                     2000       ;    # the simulation steps
--time-step                   5             ;    # the integration time step (fs)
--electrostatic               debye-huckel  ;    # the electrostatic interactions
--debye-huckel-dielectric     85            ;    # the value of relative dielectric constant␣
↪for debye-huckel equation
--debye-huckel-temperature    300           ;    # the temperature for debye-huckel equation
--debye-huckel-ionic-strength 0.00015       ;    # the ionic-strength value for debye-huckel␣
↪equation
--ele-alpha                   0             ;    # the alpha parameter for ZMM method
--thermostat                  none               # options for using thermostat in MD
--temperature                 300           ;    # simulation temperature
--com-motion                  cancel      ;      # the option for canceling the motion of␣
↪center-of-mass (COM)
--com-cancel-group-name       all                # the name of predefined group for the␣
↪canceling of COM motion
--group-o-coord    all                           # the name of predefined group to output the␣
↪trajectory
--print-interval-log          100           ;    # the interval steps of making logs
--print-interval-coord        100          ;   # the interval steps of making cods
--fn-o-coord                  md.cod        ;    # the name of the trajectory output file
--format-o-coord              presto             # the file format for the trajectory (only
↪"presto" is supported currently)
--fn-o-restart                md.restart         # the file contains the final conformation's␣
↪positions
--nsgrid-cutoff               23                 # the threshhold distance for neighbor␣
↪molecules
--nsgrid-update-intvl         5                # the update interval for nsgrid
--hydrophobicity-scale-epsiron 0.2               # a parameter for HPS model
--nonbond hydrophobicity-scale-lj                # indication of using Lennerd-Jones potential
--expected-num-density        0.1                #  a parameter to define the memory size. It␣
↪is not recommended to change this default value.
```

5.1.6 Execute omegagene

To run an MD simulation using myPresto/omegagene, execute the following command. then please wait untill the job is done.

```
${OMEGABIN}  --cfg md.inp.run --inp md.inp.cls > md.out
```

Simulation log is given in *md.out*, and the trajectory is *md.cod*.

5.1.7 Visualize the resulant trajectory

The trajectory file md.cod is written in myPresto format. This can be converted into the Gromacs trajectory .trr format.

```
python2.7 ${OMEGATK}/trajconv_presto_gro.py --i-pdb inp.pdb --i-crd md.cod -o md.trr --lx 50 --
↪ly 50 --lz 50
```

The trajectory can be visualized by some standard visualizers (e.g., VMD and PyMOL).

In addition, the final snapshot in the restart file can be converted into .pdb file format

```
python2.7 ${OMEGATK}/restart_to_pdb.py -i md.restart --i-pdb inp.pdb -o finalstep.pdb
```

5.2 VcMD

5.2.1 Files

The directory *$ {PROJECT_ROOT}/samples/cg_q8_vcmd* in the repository is used for the VcMD tutorial. In addition, files in *$ {PROJECT_ROOT}/samples/cg_q8* are also used. These two directories should be copied into your working directory.

In *$ {PROJECT_ROOT}/samples/cg_q8_vcmd* directory, there are directories named as *1* and *2*. They correspond to the first, and second iterations. In each directory, 10 parallel simulations will be carried out in directories named "n1", "n2", ..., "n10".

```
cd ${PATH_TO_YOUR_WORKIND_DIRECTORY}
cp -r ${PROJECT_ROOT}/saples/cg_q8 .
cp -r ${PROJECT_ROOT}/saples/cg_q8_vcmd .
```

5.2.2 The first iteration

In *1* directory, execute the following scripts attached to the samples. Note that these scripts requiers appropriate settings for ${OMEGABIN} and ${OMEGATK} to your omegagene binary and toolkit directory.

```
cd ${PATH_TO_YOUR_WORKING_DIRECTORY}/cg_q8_vcmd/1
$ bash c1_gen_inp.bash
```

This script generates the directory *n1* to *n10*.

```
$ bash c2_exe.bash
```

This script sequentially execute simulations from *n1* to *n10*.

```
$ bash c3_prep_next.bash
```

This script performes postprocessing for an iteraction.

- *vcmd_next.inp* and *vcmd_next_qraw.dat* are generated.
- *vcmd_next.inp* describes the canonical probability for each virtual state as an input for the next iteration.
- *vcmd_next_qraw.inp* describes the probability in the entire VcMD ensemble for each virtual state.

vcmd_next_qraw.dat:

```
10
1
7 mol1 mol2
3.0 5.0
4.0 6.0
5.0 9.0
6.0 12.0
9.0 17.0
12.0 22.0
```

```
17.0 30.0
0 0.0
2 0.00152933698463
3 0.0271170564086
4 0.0949144766083
5 0.18308553295
6 0.316262109242
7 0.377091487806
END
```

- The first line indicates the interval steps for virtual state transition trials.
- The second line indicates the number or reaction coordinates (N_RC).
- The third line indicates the number of virtual state and name of atom groups to define the reaction coordinate.
- The following seven lines are the range of lambda for each virtual state.
- The following lines describes the sampled probability for each state.

5.2.3 The second iteration and further

In the *2* directory, the same three script should be executed. Afther that, make the directory *3* and copy files from *2* to *3*. Then, repeat the same protocols.

5.2.4 Production run

After the convergence of the distribution in *vcmd_next_qraw.dat*, execute the production run with the same manner.

5.2.5 Post-processing

Following script performs re-weighting of the ensemble. Execute it in the directory of each run (e.g., *n1*, *n2*, ... directories). Note that ${PREV_STAGE} indicates the number of previous iteration.

```
$ python ${OMEGATK}/assign_traj_vs_lambda.py
   --i-qcano ../../${PREV_STAGE}/vcmd_next.inp \
   --i-cod md.cod \
  --interval-cod 1000 \
   --i-lmb lambda.out \
  --interval-lmb 1 \
   --i-vs ttp_vcmd.out \
   --interval-vs 10  \
   -o prob.dat \
```

- *–i-qcano vcmd_next.inp* is the parameter file used for the VcMD simulation.
- *–i-cod md.cod* is the trajectory file obtained from the VcMD simulation.
- *–interval-cod 1000* should specifies the value same as the *–print-interval-coord* in *md.inp.run* file.
- *–i-lmb lambda.out* is the trajectory file for lambda values obtained from the VcMD simulation.
- *–interval-lmb 1* should specifies the value same as the *–print-interval-extended-lambda* in *md.inp.run* file.
- *–i-vs ttp_vcmd.out* is the trajectory for the virtual state obtained from the VcMD simulation.
- *–interval-vs 10* is the interval for virtual state transitions.
- *-o prob.dat* is the output file describing probabilistic weight in the canonical ensemble for each snapshot. This file is a tab-separated acsii file. The first column is the step number, the second is the probabilistic weight, the third is lambda, and tha fourth is the virtual state ID.

5.3 tutorial for multi-canonical MD (McMD) simulations

All the data required for this tutorial is in the directory "samples/mcmd_ala3". In this document, ${WORKING_DIR} indicates the path to the workind directory for this simulations.

All the files should be copied in to ${WORKIND_DIR}

```
$ cp -r samples/mcmd_ala3 ${WORKIND_DIR}
```

5.3.1 cal01_inp

- ala3.pdb ... The initial structure.
- ala3.shk ... SHAKE definition file.
- ala3.tpl ... Topology file.
- atom_groups.inp ... Atom group definition file.

This sample system is consisting of three Ace-Ala-Nme peptide molecules surrounded by water molecules.

5.3.2 cal02_nvt

NVT simulations to prepare initial conformaions of McMD. In this directoly, 10 successive NVT simulations will be performed. First, the initial atomic velocities are generated by the following command.

```
$python2.7 ${OMEGATK}/presto_generate_velocities.py
  -i ../cal01_inp/ala3.pdb
  --i-tpl ../cal01_inp/ala3.tpl
  -t 600
  -o run00000_o.restartf
  -s ${RANDOM}
  —mol --i-shk ../cal01_inp/ala3.shk
```

${RANDOM} means the random seed. Define the variable ${RANDOM} or replace it into digits.

Then, simulations are performed by executing *run.bash*. Variables *OMEGAROOT*, *OMEGABIN*, and *OMEGATK* should be changed depending on your environment.

- OMEGAROOT ... the install directory of omegagene.
- OMEGABIN ... the executable binary of omegagene.
- OMEGATK ... the toolkit directory of omegagene.

run.inp and *run.system* are the templates of input files for each simulation. run00001.inp, run00002.inp, ..., and run00001.system, run00002.system, ... will be generated by *run.bash*.

After that, restart files recording the final snap shot of each of the 10 runs are converted into .pdb files.

```
$bash rst_to_pdb.bash
```

cal03_mcmd_inp

_kkmcmdconf.txt file is the configuration file for the McMD.

```
--mode               celeste
--job-script         ${PRJ_HOME}/cal03_mcmd_inp/job.bash
--project-log        ${PRJ_HOME}/kkmcmd.log
--filename-stem      md
--project-name       ala
--project-home       ${WORKING_DIR}
--n-digit-run-id     5
--cell-size          39.7098 39.7098 39.7098
```

```
--random-seed        16102812
--inp-topology       ${PRJ_HOME}/cal01_inp/ala3.tpl
--inp-shake          ${PRJ_HOME}/cal01_inp/ala3.shk
--inp-ttp            ttp_v_mcmd.inp
--out-ttp            ttp_v_mcmd.out
--inp-vert           start.vert
--out-mc-ene         mult.ene
--n-trivial-parallel-mpi 1
--job-script-mpi  ${PRJ_HOME}/cal03_mcmd_inp/job_mpi.bash

; settings phase=0
--cal-dir        ${PRJ_HOME}/cal03_mcmd_inp

; prerun phase=1
--cal-dir        ${PRJ_HOME}/cal04_mcmd1
--prerun-init        ${PRJ_HOME}/cal02_nvt/run 1 1 10
--prerun-force-coef  0.8  0.1  10
--prerun-md-init-template    ${PRJ_HOME}/cal04_mcmd1/md.inp
--prerun-md-inp-template    ${PRJ_HOME}/cal04_mcmd1/md.inp.run
--prerun-ttp-inp-template   ${PRJ_HOME}/cal04_mcmd1/pre_ttp.inp.tmpl
```

File paths and cell sizes should be adjusted depending on you environment and you system.

The lines under ”; prerun phase=1” set configulation of pre-run of McMD performed in the *cal04_mcmd1* directory. In this phase, 10 succsessive runs are performed for 10 independent series of runs (10 * 10 = 100 simulations are performed). During the successive runs, the strength of biasing potential is gradually weakens.

The file *job.bash* is a shell script for execution of omegagene. Edit this file to specifies the path to the binary of omegagene.

5.3.3 cal04_mcmd1

The following files are input configuration files.

- md.inp
- md.inp.run
- pre_ttp.inp.tmpl

The following command generates inputs of 10 independent runs.

```
$ alias mcmd="python cal03_mcmd_inp/kkmcmd_job_control.py -i cal03_mcmd_inp/_kkmcmdconf.txt"
$ mcmd
```

This command will be repeatedly executed during the iterations of McMD simulations.

In *cal04_mcmd1/1* directory, n1, n2, ..., n10 are generated.

Then, execute *job.bash* in each of n*X* directories. After ending of all the 10 runs, execute *mcmd* command twice.

```
$ mcmd
$ mcmd
```

In *cal04_mcmd1/2* directory, n1, n2, ..., n10 are generated.

Repeat this process 10 times.

Finally, execute the script *gen_ttpvout.bash*.

```
$ bash gen_ttpvout.bash
```

5.3.4 for_next ; postprocess of cal04_mcmd1

After ending the *cal04_mcmd1/10*, the initial guess of the density of states is calculated by using *for_next* directory. Edit the text file *for_next/current_situation*.

```
4
1
1
60
```

The first line indicates the phase; 4 indicates cal04_mcmd1. The second line is the number of virtual states. This value is 1 for cal04_mcmd1. The remaining two lines are not used in the current version.

Next move to the directory *for_next/v_distrib/cal04_mcmd1_pre*. Excecute the following command.

```
$ bash com.bash 10 10 5
```

Note the minimum and maximum energies.

```
$ tail -n1 stg_1/v_pdf/s1.pdf
-1.4130000e+04   -1.2206073e+01
$ head -n1 stg_10/v_pdf/s1.pdf
-1.9970000e+04   -1.2899220e+01
```

The first value is the energy (kcal/mol), and the second value is log probability.

Edit the text file *range.info* to input these minimum and maximum energeis at the first line.

```
$ python2.7 gen_mcmdinp.py -i range.info -o ttp_v_mcmd.inp
$ cp ttp_v_mcmd.inp ../../../cal04_mcmd1
```

Execute the following program again.

```
$ bash com.bash 10 10 5
```

Then, move to the directory *derv_den_Pc*. Execute the command

```
$ cd ../../derv_de_Pc
$ csh com
```

When the follwing text appears, input “1” by your keyboard.

```
nstage =  10
####  If OK, input 1. ####
```

If ifort is not in your environment, “ifort: Command not found.” appears. Edit *com* to replace ifort into your fortran compiler.

Move to the directory *fit_dden*, Edit the *cal04_mcmd1/inp.dat_e1* to specify the minimum and maximum potential energies as the same way for *range.info* which was editted above.

```
$ emacs -nw  cal04_mcmd1/inp.dat_e1
```

```
-19970.0   5.0    999.0  0.0 3 0.0
-19970.0  -14130.0   -19970.0  -14130.0
-19970.0  -14130.0
```

Execute the command

```
$ csh com_pre
```

When the follwing text appears, input “1” by your keyboard.

```
This is specialized for md1_*.  Are you OK?

####  If OK, input 1. ####
```

The file *e1_fort.20* will be generated. This file describes the parameters for McMD. By using this file, the input files for the next iteration will be prepared.

5.3.5 cal05_mcmd1

The directory for the next iteration *cal05_mcmd1* should be copied from the *samples/mcmd_ala3* directory of the omegagene repository.

```
$ cd ../../
$ cp OMEGAGENE_REPOSITORY_DIR/samples/mcmd_ala3/cal05_mcmd1 . -r
```

The McMD parameter file *cal05_mcmd1/ttp_v_mcmd.inp* is prepared by adding the content of *e1_fort.20* to the tail of *cal04_mcmd1/ttp_v_mcmd.inp*. See the sample file, *samples/mcmd_ala3/cal05_mcmd1/ttp_v_mcmd.inp*. In this file, there are nine copies of the content of *e1_fort.20*, because this sample simulation configured to use the nine virtual states.

The next iteration is defined in *cal03_mcmd_inp/_kkmcmdconf.txt*

```
; cal05_mcmd1 phase=2
--cal-dir        ${PRJ_HOME}/cal05_mcmd1
--mcmd-stages              2  1
--mcmd-inp-ttp             2 ${PRJ_HOME}/cal05_mcmd1/ttp_v_mcmd.inp
--mcmd-md-inp-template     2 ${PRJ_HOME}/cal05_mcmd1/md.inp.run
--mcmd-md-init-template    2 ${PRJ_HOME}/cal05_mcmd1/md.inp
--mcmd-init                2  1:1:1     1:1:2    1:1:3   1:1:4    1:1:5
--mcmd-init                2  1:1:6     1:1:7    1:1:8   1:1:9   1:1:10
```

To run the simulation, execute the *mcmd* command.

```
$ mcmd
$ mcmd
```

Then, execute *job.bash* in each of cal05_mcme1/1/nX directories, where X is one of 1-10.

5.3.6 Postprocessing of cal05_mcmd1

After finishing the simulations, post-processing will be done in *for_next* directory.

```
$ cd for_next
```

Edit the first two lines of *current_situation* as follows

```
5
9
1
60
```

```
$ cd v_distrib/cal05_mcmd1
```

Edit the *com_pre.bash* to set the variable OMEGATK to your omegagene toolekit directory.

```
$ OMEGATK=${HOME}/local/og0/toolkit
```

Then, execute the script.

```
$ csh ./com_pre.bash 1 10
```

Argument 1 and 10 indicates the number of runs. The histogram of populations for each potential energy bin for each virtual state is generated in *v_pdf* directory. If *R* is working, the image file *v_distrib.png* is generated.

After that, Run the following scripts.

```
$ cd ../../fit_pmc_entire
$ csh ./1234_com 4 7 0
```

```
$ cd ../
$ csh do_fitmix_nextpre.csh 7 0
```

The directory *cal06_mcmd1* will be generated in the parent directory. Run this iteration in the same way as the previous iteration.

Repeat the iterations till the converge the distribution obtained in *v_distrib* directory. From the *cal06_mcmd1*, use *do_fitmix_iter.csh* instead of *do_fitmix_nextpre.csh*.

After convergence of the distribution, perform a production run.

Reweigthing of the production run is performed *for_next/gen_p_cano_McMD* directory. Edit *md_vst* file as follows.

```
14
9
5.0
```

The first line is the number of iterations. If the production run is *cal15_mcmd1*, input *14* here. The second line is the number of virtual states. The third line is bin for the potential energy in kcal/mol unit.

```
$ csh 1_com
$ csh 2_com_integ
$ csh 3_com_P_E_T
```

Then, the potential energies of each snapshot in the trajectory are obtained by the following script.

```
$ python2.7 ${OMEGATK}/kkmcmd_pot_from_crd.py \
    --cod md.cod \
    -o    pot.txt
```

- *md.cod* is a trajectory file generated by omegagene.
- *pot.txt* is output file.

The probability of existance in the canonical ensemble is obtained by the following scripts.

```
$ echo pot.txt > list.txt
$ python2.7 ${OMEGATK}/kkmcmd_reweighting.py \
  --flg-pot -i pot.txt \
  --i-cano for_next/gen_p_cano_McMD/p_cano/P_E_T300.dat \
  -o prob.txt
```

5.3.7 For the AUS method

The protocol for the AUS method is similar to the McMD. For running an AUS simulation on the omegagene, following settings are required.

- –fn-i-aus-restart ... A file name for the output restart file.
- –aus-type ... This should be “dist-mass-center*.
- –enhance-group-name ... Names for two atom groups should be specified. The distance between centroids of these groups is used as the reaction coordinate.

The same protocol using *for_next* scripts can be applied to the AUS method.