CafeMol 2.0 manual

Hiroo Kenzaki,
Nobuyasu Koga, Naoto Hori, Ryo Kanada,
Wenfei Li, Kei-ichi Okazaki, Tsuyoshi Terakawa, Xin-Qiu Yao, and
Shoji Takada

Department of Biophysics, Graduate School of Science, Kyoto University,
Department of Physics, Nanjing University

May 31, 2012

© 2009-2012 Kyoto University
## Contents

0.1 Changes from version 1.0 .................................................. 7  
  0.1.1 Update from version 1.0 ........................................... 7  
0.2 Contributors .............................................................. 7  
0.3 License ..................................................................... 8  
0.4 Bug reports ................................................................. 8  
0.5 Acknowledgement .......................................................... 8  
  0.5.1 Copyright of Multiple stream Mersenne Twister .................. 8  

1 Introduction .................................................................. 11  
  1.1 Concept of coarse-grained MD ....................................... 11  
  1.2 Simulating proteins at work ......................................... 13  

2 Getting started ............................................................. 15  
  2.1 About the code ......................................................... 15  
  2.2 How to install CafeMol .................................................. 15  
  2.3 How to execute CafeMol ................................................. 16  
  2.4 How to restart a simulation ......................................... 17  
  2.5 Size limit and memory resource .................................... 17  

3 Simulation methods ........................................................ 19  
  3.1 Units in CafeMol .......................................................... 19  
  3.2 Model and energy function ............................................ 19  
    3.2.1 Off-lattice Go model for proteins .............................. 19  
    3.2.2 Atomic interaction based CG (AICG) model for proteins (AICG1) ........................................... 21  
    3.2.3 AICG with the flexible local potential (AICG2) ......... 21  
    3.2.4 Flexible local potential ......................................... 22  
    3.2.5 Multiple-basin model for proteins ............................ 23  
    3.2.6 RNA model .......................................................... 25  
    3.2.7 DNA model .......................................................... 27  
    3.2.8 Elastic network model of proteins and RNAs ............. 28  
    3.2.9 Electrostatic interaction ........................................ 28  
    3.2.10 Hydrophobic interaction ...................................... 29  
    3.2.11 Excluded volume interaction ................................. 30
4.1 Input files ............................................ 39
  4.1.1 PDB file (.pdb) .................................. 39
  4.1.2 Native-info file(.ninfo) .................................. 39
  4.1.3 Input file (.inp) .................................. 44
  4.1.4 Parameter file (.para) .................................. 44
  4.1.5 AICG files ........................................ 49
4.2 Output files ............................................ 50
  4.2.1 PDB file (.pdb) .................................. 50
  4.2.2 CARD coordinate (.crd) and velocity (.velo) .................. 50
  4.2.3 DCD coordinate (.dcd) and velocity (.vdcd) .................. 51
  4.2.4 PSF (protein structure file) file (.psf) .................. 53
  4.2.5 Movie file (.movie) .................................. 53
  4.2.6 Data file (.data) .................................. 53
  4.2.7 Time-series file (.ts) .................................. 54
  4.2.8 Native-info file(.ninfo) .................................. 56
  4.2.9 Replica information file(.rep) ............................ 56
5 Input file: How to make

5.1 General rules of input files ............................................. 59
5.2 <<<< filenames (required) ............................................. 60
5.3 <<<< job_crtl (required) ............................................... 62
5.4 <<<< unit_and_state (required) ....................................... 63
5.5 <<<< native_info_simN (N can be 1, 2,..)(optional) ............... 65
  5.5.1 All-in-one-file style ............................................. 65
  5.5.2 One-by-one-file style ............................................ 65
5.6 <<<< initial_struct (optional) ...................................... 66
5.7 <<<< initial_velo (optional) ......................................... 67
5.8 <<<< initial_lipid (not implemented) ................................ 67
5.9 <<<< sequence (optional) ............................................ 67
5.10 <<<< dssp_file (optional) ........................................... 67
5.11 <<<< energy_function (required) ................................... 68
5.12 <<<< flexible_local (optional) ..................................... 71
5.13 <<<< aicg (optional) .................................................. 72
5.14 <<<< multiple_go (optional) ....................................... 72
5.15 <<<< electrostatic (optional) ....................................... 73
5.16 <<<< hydrophobic (optional) ....................................... 74
5.17 <<<< md_information (required) ................................... 75
5.18 <<<< mpc_dynamics (optional) ..................................... 79
5.19 <<<< implicit_ligand (optional) ................................... 80
5.20 <<<< binding_site (optional) ...................................... 81
5.21 <<<< redefine_para (optional) .................................... 82
5.22 <<<< energy_para (optional) ....................................... 82
5.23 <<<< neighbor_dist (optional) .................................... 83
5.24 <<<< mass_friclion (optional) ] ................................... 83
5.25 <<<< del_interaction (optional) ................................... 83
5.26 <<<< anchor_para (optional) ....................................... 84
5.27 <<<< bridge_para (optional) ....................................... 84
5.28 <<<< pulling_para (optional) ..................................... 85
5.29 <<<< fix_para (optional) ............................................ 85
5.30 <<<< in_box (optional) ............................................... 85
5.31 <<<< searching_tf (optional) ...................................... 86
5.32 <<<< annealing (optional) .......................................... 86
5.33 <<<< replica (optional) .............................................. 86
5.34 <<<< replica_XXXX (optional) ..................................... 88
5.35 <<<< modified_muca (optional) .................................... 89
6 Examples: Input and output

6.1 Protein native dynamics and (un)folding with Go model at constant $T$ ........................................ 91
   6.1.1 Input & Output ........................................ 91

6.2 Protein native dynamics and (un)folding with AICG model at constant $T$ ........................................ 91
   6.2.1 Input & Output ........................................ 91

6.3 Automatic $T_F$ search ........................................ 92
   6.3.1 Input & Output ........................................ 92

6.4 Conformational transition simulation with multiple basin model at constant temperature ...................... 92
   6.4.1 Input & Output ........................................ 92

6.5 Conformational transition simulation with multiple basin model and implicit-ligand (MD-MC scheme) .......... 92
   6.5.1 Input & Output ........................................ 92

6.6 Cyclic conformational change simulation by switching Go model at constant temperature ...................... 93
   6.6.1 Input & Output ........................................ 93

6.7 Rotary motion of F$_1$-ATPase by switching Go model at constant temperature .................................... 93
   6.7.1 Input & Output ........................................ 93

6.8 Conformational change and an explicit ligand(drug) dissociation in a multidrug transporter by multiple-basin model at constant temperature ........................................ 93
   6.8.1 Input & Output ........................................ 94

6.9 Protein-protein docking with electrostatic interactions ................................................................. 94
   6.9.1 Input & Output ........................................ 94

6.10 Temperature replica exchange molecular dynamics simulations ......................................................... 94
   6.10.1 Input & Output ........................................ 94

6.11 tRNA ................................................................ 95
   6.11.1 Input & Output ........................................ 95

6.12 DNA ................................................................ 95
   6.12.1 Input & Output ........................................ 95

6.13 Using coarse-grained representation as an initial structure .............................................................. 95
   6.13.1 Input & Output ........................................ 95

6.14 p53 searching its specific binding site on DNA simulated using flexible local potential ....................... 95
   6.14.1 Input & Output ........................................ 95

6.15 Reversible dissociation and association of a protein hetero-dimer with MPC dynamics ......................... 96
   6.15.1 Input & Output ........................................ 96

7 Utilities: How to use ........................................ 97

7.1 Post-MD analysis ........................................ 97
   7.1.1 Calculation of RMSD ..................................... 97

7.2 Data conversion ........................................ 98
   7.2.1 show rst and a2rst ..................................... 98

Bibliography ........................................ 99
Preface

CafeMol is a general-purpose coarse-grained (CG) molecular dynamics (MD) simulation software for biomolecular systems (Kenzaki et al., 2011). It can simulate proteins, nucleic acids, lipid membrane and their complexes with various CG models. (The lipid part is not yet released.)

0.1 Changes from version 1.0

Major changes from version 1.0 to 2.0 are the followings:

1. A CG model for DNA is now available.
2. A new CG model for RNA is implemented.
3. For proteins, an updated version of the AICG model, designated as AICG2, is implemented.
4. For proteins, a statistical local potential, called the flexible local potential, is implemented.
5. Multi-particle collision (MPC) dynamics is implemented.
6. A modified multi-canonical ensemble method is now available.
7. Input style for the energy function is renewed.
8. “Restart” function is now supported.
9. Many bugs are fixed (and created!).

0.1.1 Update from version 1.0

In order for the users of the older version to update the CafeMol version, one has to renew all components of CafeMol. Not only the source code, but also parameter directory and so forth were changed.

We note the change from version 1.0 to 2.0 is not necessarily upward compatible although we tried to minimize incompatibility. Some input files prepared for the version 1 may not work in the version 2.

0.2 Contributors

The main contributor of CafeMol is Hiroo Kenzaki. The chief contributor for the earlier version of CafeMol, called CaFold (Koga and Takada, 2001), is Nobuyasu Koga. Other key contributors of CafeMol are (in alphabetical order), Naoto Hori, Ryo Kanada, Wenfei Li, Kei-ichi Okazaki, Tsuyoshi Terakawa, and Xin-Qiu Yao. The CafeMol project is directed by Shoji Takada.

Contact address is cafemol@theory.biophys.kyoto-u.ac.jp.
0.3 License

*CafeMol* is a non-commercial software that users can use, modify, and improve the *CafeMol* source code at one's own expense. The source code of *CafeMol* is available for the internal use only at this moment: The user shall not distribute or transfer *CafeMol*, and any modifications, improvements, or derivatives to *CafeMol* that the user may create.

*CafeMol* is a research tool still in the development stage, that is being supplied "as is," without any accompanying services or improvements from developers and the program is distributed to enable users to utilize *CafeMol* in their researches. Developers accept no obligation to provide maintenance nor does it guarantee expected functionality of *CafeMol* or of any part of *CafeMol*.

Any published work which utilizes *CafeMol*, or any part of *CafeMol*, must include the citation of (Kenzaki et al., 2011).

0.4 Bug reports

*CafeMol* is an infant program, most likely with many bugs. We are of course eager to remove them as much as possible. Thus, we strongly ask users to report bugs they find. Contact address is cafemol@theory.biophys.kyoto-u.ac.jp.

0.5 Acknowledgement

*CafeMol* development has been supported by Research and Development of the Next-Generation Integrated Simulation of Living Matter, a part of the Development and Use of the Next-Generation Supercomputer Project of the Ministry of Education, Culture, Sports, Science and Technology.

*CafeMol* contains a set of IO utility subroutines, which was modified from a set of tools, UTILKOTO, originally coded by Shigeru Obara.

*CafeMol* utilizes random number generator, "Mersenne twister", in two forms. One is the routine originally created by Drs. Makoto Matsumoto and Takaji Nishimura, and modified by Drs. Hiroshi Takano and Richard Woloshyn. The routine "mtmod.F90" is a library, not a part of *CafeMol*, is distributed under LGPL. Another is "Multiple stream Mersenne Twister" created by Ken-Ichi Ishikawa, which is distributed with New BSD License. The copyright information for this is given in the following.

0.5.1 Copyright of Multiple stream Mersenne Twister

Copyright (c) 2010, Ken-Ichi Ishikawa [ishikawa[at]theo.phys.sci.hiroshima-u.ac.jp] All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.

* Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Chapter 1

Introduction

*CafeMol* is a general-purpose coarse-grained (CG) molecular dynamics (MD) simulation software for biomolecular systems. It can simulate proteins, nucleic acids, lipid membrane, and their complexes with various CG models. (At this stage, though, the lipid part is not publicly available yet)

1.1 Concept of coarse-grained MD

Molecular dynamics (MD) simulation of biomolecules has long been used for a variety of studies in biology at molecular level. Majority of them employ all-atom representation with molecular mechanics force fields and classical mechanics as the principle of dynamics. Albeit their enormous success, all-atom MDs still have serious limitation in time scale. All-atom MD reaches up to at best millisecond in a specific environment and more commonly up to microseconds, but typical biological phenomena take milliseconds, seconds, or longer. Thus, most of biologically relevant events cannot be simulated directly by all-atom MDs, unfortunately. In such a situation, a strategy of coarse-graining the molecular representation has been taken in many occasions (Takada, 2012). *CafeMol* is developed on this line. By coarse-graining, we can easily simulate biological events that correspond to much longer time scale. Such coarse-grained (CG) models are designed often based on some knowledge of all-atom models. Multiscale simulation methods often provide us methods to determine parameters in CG models based on all-atom models.

Coarse graining is not a unique procedure at all. Each CG model is based on its developer’s perspectives. *CafeMol* takes one particular way of coarse-graining, which is based on specific perspectives on biomolecular system. Here, we briefly describe it.

Many proteins have evolved to have ability of folding to their own native structures. If we look at native structures of proteins in PDB, we notice that, overall, most side-chains are extremely well-packed in their cores. Whenever you find a charge in the core, it is either paired with counter charges or functionally essential. Thus, except functional reasons, the interaction at the native is highly consistent, which was termed many years ago as “consistency principle” by Go (Go, 1983). Proteins gained, through evolution, the foldability by minimizing the frustration at their native structures, which is called “principle of the minimum frustration” by Wolynes (Bryngelson and Wolynes, 1987). The effective energy takes the minimal value at the native, and as the conformation deviates from the native, the energy, on average, increases, which leads to funnel-like picture of the energy landscape, first invented by Onuchic (Leopold et al., 1992). We emphasize that completeness of the side-chains packing are outstanding, in particular. In this sense, the native state is of some similarity to crystal. On the other hand, the denatured state is characterized with low-level side-chain packing and large fluctuation and thus resembles fluid. Coarse-graining is relatively easy for fluid phase because it is essentially statistical ensemble that decides the property of polymers. All you need is statistical average over the ensemble, which makes coarse-graining easier. The native state, however, is more difficult to be described by CG model because it is defined by high level of side-chain packing, a very specific and non-self-averaging property. If the side-chain architecture is lost by coarse-graining, very surely you lose those specific interactions, to some extent. Thus, purely physics based coarse-graining is not very effective for approximating the native state. This is the stage we should bring the evolutionary perspective. Remember that we know that proteins have evolved to have the minimal frustration. This
principle can be used as a guiding principle for coarse-graining. Namely, we assume, as an extreme, that all the interactions found at the native structure are attractive. Simultaneously, we require that the protein can take nearly random coil at sufficiently high temperature. These two extreme requirements lead to the so-called Go model, first developed in the lattice representation of proteins. However, of course, protein dynamics near the native is not well approximated by the lattice representation. Indeed, protein dynamics near the native structure may be well-approximated by quasi-harmonic potential, such as elastic network model (ENM) of Tirion (Tirion, 1996). The ENM model is good only near the native and is not appropriate for the denatured state. A model that is similar to the ENM near the native structure and simultaneously that share the concept of the lattice Go model is required. This led to the idea of off-lattice Go model. Off-lattice Go model represents quasi-harmonic fluctuation near the native structure and simultaneously realize perfect funnel energy landscape. So, Go model is often called the perfect-funnel model, the native-centric model, or structure-based model. CafeMol employs off-lattice Go model developed by Clementi, Nyemyer, and Onuchic (Clementi et al., 2000), and its derivative, as a basic CG model for proteins.

We also note that, by emphasizing the structure-based potential, we are not completely ignoring physico-chemical interactions of proteins. For some purposes, those physico-chemical interactions are indispensable. One clear example is modeling of intrinsically disordered proteins/regions of proteins, for which no crystallographic structure is available. Some generic physico-chemical potentials, both in local potential and in non-local potential, can be turned on in CafeMol. Even within specific and native-centric interactions, all the interaction strengths should not be identical; some stronger than others. In CafeMol, some CG models try to incorporate those physico-chemical interactions as much as possible.

In contrast, lipid membrane is fluid-like. It is well-known that lipids can diffuse laterally in lipid membrane, which directly suggest fluidity. Fluctuation on vertical direction is also noticeable. Thus, physics-based coarse-graining is expected to work better for lipid membrane. Indeed, transferable CG models of membrane have been developed and widely used. It is also noted that lipids are much shorter polymer than proteins and lipids themselves are not explicitly encoded in genome. Thus no direct evolutionary pressure exists so that lipid take a specific structure. Structure-based model, or Go model is not appropriate. Thus, Go model is not used at all for lipid membrane in CafeMol.

Nucleic acids are in between proteins and lipids, as for the degree of evolutionary design. Some RNAs work as ribozyme, for which structure-based model description is perhaps the best choice because the base sequences for these RNA must have evolved to have the foldability. Some other RNAs do not have foldability, and thus more generic potentials are necessary. DNA makes duplex in most situations, and thus Go-like description is useful to support the B-DNA duplex shape. Yet, since nucleic acids always contain a lot of negative charges, electrostatic interactions play major roles in every situations, and thus physical interaction energy is also of crucial importance. CafeMol employs CG model of duplex DNA that combine Go-like interaction and physical interactions (Knotts et al., 2007; Sambriski and de Pablo, 2009).

Perhaps, the most difficult decision is about the interaction between different types of biomolecules. Candidates are physical interactions, Go interactions, and both of them. We recommend to classify inter-molecular interactions into two types: (sequence-) specific and generic. The specific interaction, by definition, is specific to the particular pair of molecules, and thus is sequence dependent. It is also characterized by relatively high affinity. Thus, removal of side-chain architecture in the CG model lose the van der Waals packing and, most likely, destabilizes the specific interaction. Simultaneously, it can be thought that the specific interaction was designed by the evolutionary pressure. They together legitimate the use of Go interaction for the specific interaction. (There is no problem to add physical interaction, together with Go interaction, to the specific interaction) On the other hand, generic interaction is often weak and, by definition, does not sensitively depend on the sequence. Thus, coarse-graining may not destabilize the generic potential very seriously. Thus, we recommend to use physico-chemical potentials for the generic interactions. For example, interactions between proteins and lipid membranes are generic. Protein-DNA interactions depend on the cases. The complex of transcription factor and DNA is best modeled by both Go and physical interactions. Nucleosome may be modeled by physical interactions between histone proteins and duplex DNA.

As a generic software of CGMD, CafeMol implements both structure-based Go potentials and generic physico-chemical potentials, and how to use them is users’ choice.
1.2 Simulating proteins at work

Biomolecules, primarily proteins and some RNA’s work as biomolecular machines. To understand the working principles of such machines, molecular dynamics simulations are potentially very powerful because they provide time-dependent structural information. By using CG model, we can simulate the protein dynamics in a time scale comparable to that the biomolecular machines indeed work. Yet, the long-time simulation alone is not sufficient to simulate such machines at work, because proteins actively move (work) driven by some external energy source. Often, the energy source is of chemical nature, e.g., chemical reaction, ion passage, and so on. CG MD cannot directly deal with such chemical events. For mimicking these energy source, we proposed to “switch” the energy function in certain ways (Koga and Takada, 2006). By switching, we put some energy into the system. Proteins start to “work” as machines.

One of the key advantages of CafeMol, in comparison with other MD packages, is to equip various means to conduct dynamic “switching” simulations, which try to mimic energy source given into the system and switch on the active motion of the machines. Molecular motors such as F$_1$-ATPase (Koga and Takada, 2006) and kinesin are wonderful examples of these functions.
Chapter 2

Getting started

2.1 About the code

*CafeMol* is written in Fortran90/95 with MPI, OpenMP and C pre-processing directives. Thus, it can run virtually any computer that has both Fortran90/95 and C compiler. (MPI and OpenMP can be disabled by using some pre-processor directives.) In an ordinary way, files having extension ‘.F90’ are automatically and firstly pre-processed by C compiler.

2.2 How to install *CafeMol*

The following steps are typically needed.

1. Download *CafeMol_xxx.tar.gz* file, and extract it.

   $tar zxvf CafeMol_xxx.tar.gz

   Then, the extracted directories should be as below.

   ```
   cafemol/ src/ Makefile
   SubroutineList.txt
   program source files
   para/ parameter files
   aicg/ aicg-related files
   pdb/ sample PDB files
   ninfo/ sample ninfo files
   example/ sample input and data files
   ```

2. Edit the “Makefile” as required to comply with your system. The original one is written for Intel Fortran compiler.

   $cd cafemol/src/
   $vi Makefile (or by using any editor that you like)

   For example, to modify it for GNU compiler; “gfortran”,

   (a) Comment the following lines out.

   ```
   #------ intel (single)
   FC = ifort
   FC_UTIL = ifort
   FLAG = -O2 -i-static
   #------ intel (single)
   #FC = ifort
   #FC_UTIL = ifort
   #FLAG = -O2 -i-static
   ```
(b) Then, uncomment the following lines.

```
#------ gfortran  #------ gfortran
#FC = gfortran    FC = gfortran
#FC_UTIL = gfortran FC_UTIL = gfortran
#FLAG = -O2 -static FLAG = -O2 -static
```

Lines for some typical cases; GNU and Intel compiler are prepared. If you can not find suitable lines for your system, write or edit following three items in `Makefile`.

| FC= | compiler command for generating CafeMol execution file. |
| FLAG= | compile option for generating CafeMol execution file. |
| FC_UTIL= | compiler command for generating accessory execution file. |

3. Compile.

```
$ make clean
$ make
```

which produces the execution file “cafemol” at the CafeMol home directory.

### 2.3 How to execute CafeMol

1. Place an input file (e.g. cafemol_go_1chain.inp) at the CafeMol home `./cafemol`, parameter files at `./cafemol/para/` (by default this is done), and required PDB files at `./cafemol/pdb/` for reference structures. See Chapters 4 & 5 for the input file format and contents.

2. Change the current directory to the CafeMol home `cafemol/`,

```
$ cd cafemol
```

and run the execution file,

```
$. ./cafemol example/sh3/sh3.inp
```

If you use MPI, command should be

```
$mpirun -n [number of processors] ./cafemol example/sh3//sh3.inp
```

(This command style is dependend on your MPI-running system.)

3. CafeMol generates output files as below. Only “.data”, “.ninfo”, and “.ts” are outputs by default and the others are optional. See 5.2 for the output control and Chapter 4 for the file formats.

<table>
<thead>
<tr>
<th>Extension</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>.data</td>
<td>Text</td>
<td>simulation condition</td>
</tr>
<tr>
<td>.ninfo</td>
<td>Text</td>
<td>native(reference) state</td>
</tr>
<tr>
<td>.ts</td>
<td>Text</td>
<td>time-series</td>
</tr>
<tr>
<td>.pdb</td>
<td>Text : PDB</td>
<td>coordinates for restarting (option)</td>
</tr>
<tr>
<td>.velo</td>
<td>Text : CARD</td>
<td>velocities for restarting (option)</td>
</tr>
<tr>
<td>.crd</td>
<td>Text : CARD</td>
<td>coordinates for restarting (option)</td>
</tr>
<tr>
<td>.movie</td>
<td>Text : PDB</td>
<td>trajectory coordinates (option)</td>
</tr>
<tr>
<td>.dcd</td>
<td>Binary : DCD</td>
<td>trajectory coordinates (option, only for Intel compiler)</td>
</tr>
<tr>
<td>.vdcd</td>
<td>Binary : DCD</td>
<td>trajectory velocities (option, only for Intel compiler)</td>
</tr>
<tr>
<td>.rep</td>
<td>Text</td>
<td>replica information file</td>
</tr>
<tr>
<td>.psf</td>
<td>Text : PSF</td>
<td>protein structure file (option)</td>
</tr>
<tr>
<td>.rst</td>
<td>Binary</td>
<td>restart file (option)</td>
</tr>
</tbody>
</table>

(WARNING) In some computer systems of linux, the above simple run may result in an error with messages something like “segmentation fault”. This may be because of the limit of the stack memory size. CafeMol uses relatively large stack size. One can remove the upper limit of the stack memory size by a command such as
2.4 HOW TO RESTART A SIMULATION

2.4 How to restart a simulation

After a run, one can continue/restart the similar run using a retart file(.rst) and a set of input files(.inp, .pdb, etc.) which are identical to the previous simulation. Note that the restart file(.rst) is an optional output and thus is not produced by default. (To creat the rst file, you should specify “rst” in the “OUTPUT” line in your input file.)

To run the restart job, one needs to specify the path of the restart file as the second argument of the execution command, e.g,

$ ./cafemol ./example/hoge/hoge_2nd.inp ./example/hoge/hoge.rst

Currently, this restart procedure is not available for some sorts of simulations in CafeMol. Namely, the restart file contains only values of coordinates, velocity, and acceleration of mass points, step number, and some replica variables. Therefore one can not resume a simulation which needs other internal variables such as states of implicit-ligand molecule in the case of “i_implig = 1”. The contents of the restart file can be confirmed by using utility program, show_rst. (See 7.2.1.)

2.5 Size limit and memory resource

CafeMol's size limits of the simulating system are defined by parameters in src/const.maxsize.F90. These max-numbers also affect the required memory for execution. Among many max-numbers, you may need to watch the following several ones which most significantly change the total memory size required. If your simulation target is so large or contains more states than anticipated by default, you should increment them. On the other hand, if your computer environment has only little memory, you should diminish them.

<table>
<thead>
<tr>
<th>parameter</th>
<th>default</th>
<th>information</th>
</tr>
</thead>
<tbody>
<tr>
<td>MXMP</td>
<td>100000</td>
<td>max number of mass points</td>
</tr>
<tr>
<td>MXUNIT</td>
<td>200</td>
<td>max number of units</td>
</tr>
<tr>
<td>MXMPNEIGHBOR</td>
<td>500</td>
<td>max length of the neighbor list per number of MPs</td>
</tr>
<tr>
<td>MXMPELE</td>
<td>7000</td>
<td>max length of the electrostatics neighbor list per number of charges</td>
</tr>
<tr>
<td>MXSYSTEM_MGO</td>
<td>20</td>
<td>max number of multiple-Go systems</td>
</tr>
<tr>
<td>MXSTATE_MGO</td>
<td>3</td>
<td>max number of states in a multiple-Go system</td>
</tr>
</tbody>
</table>

where “mass points” stands for CG particles that include amino acids in proteins, phosphates, bases, and sugars in nucleic acids, and particles in lipids. “Units” correspond to molecular (chain) units which include proteins, nucleic acids, collections of all the explicit ligands, and collection of all the lipid molecules.

Regarding the memory usage, MXMP (or "nmp”; its actual number to be used) has the largest impact. In particular, the neighbor list arrays are probably the largest arrays. For excluded volume term, its size is MXMPNEIGHBOR × nmp, and for electrostatic interactions, its size is MXMPELE × (number of charges).
Chapter 3

Simulation methods

3.1 Units in CafeMol

*CafeMol* uses a specific unit which will be defined here.

1. The length unit is Å, which is thus $10^{-10}m = 10^{-1}nm$.

2. The energy unit is kcal/mol, which is $1kcal/mol = 6.9478 \times 10^{-21}J = 6.9478pN \cdot nm$. Just for your convenience, $1kcal/mol = 0.04337eV$. For temperature, we use Kelvin, $1K = 1.987 \times 10^{-3}kcal/mol$. Thus, for room temperature, $1K_B T \sim 0.6kcal/mol$.

3. The mass unit is our own one. We set that each amino acid has the mass of 10, which we call 10cafe-mu (cafemol-mass-unit). Since average mass of 20 amino acids is 137amu (atomic-mass-unit), we thus define $1cafe-mu = 13.7amu = 2.275 \times 10^{-26}kg$.

4. The unit of charge is elementary-electric charge (e); namely, one electron has the charge of $-1$.

From these, we can directly obtain the unit time of *CafeMol* to be $1cafe-time = 1.809 \times 10^{-13}s \sim 200fs$. We need to be cautious in interpreting this time scale. Since intrinsic dynamics is accelerated by coarse-graining the energy landscape, this apparent mapping in time scale does not give a good estimate of the real time.

3.2 Model and energy function

3.2.1 Off-lattice Go model for proteins

For the Go model of proteins, *CafeMol* implement a particular implementation of off-lattice Go models developed by Clementi, Nyemyer, and Onuchic (Clementi et al., 2000). The model uses CG particles where each CG particle represents an amino acid, most often represented as $C_\alpha$ atoms (can be $C_\beta$, or the center of mass of amino acids, though). For a protein with the number of amino acids $n_{aa}$, the off-lattice Go model is defined by the potential energy function,

$$V_{Go}(\mathbf{R}|\mathbf{R}_0) = \sum_{ibd} K_{b,ibd}(b_{ibd} - b_{ibd,0})^2 + \sum_{iba} K_{\theta,iba} (\theta_{iba} - \theta_{iba,0})^2$$

$$+ \sum_{idih} \left\{ K_{\phi,idih}^1 [1 - \cos (\phi_{idih} - \phi_{idih,0})] + K_{\phi,idih}^3 [1 - \cos 3(\phi_{idih} - \phi_{idih,0})] \right\}$$

$$+ \sum_{i<j=3}^{nat\ contact} \varepsilon_{go,ij} \left[ 5 \left( \frac{r_{ij0}}{r_{ij}} \right)^{12} - 6 \left( \frac{r_{ij0}}{r_{ij}} \right)^{10} \right]$$

$$+ \sum_{i<j=3}^{non-native} \varepsilon_{ev} \left( \frac{d}{r_{ij}} \right)^{12}$$

where

\begin{align*}
V_{Go}(\mathbf{R}|\mathbf{R}_0) &= \sum_{ibd} K_{b,ibd}(b_{ibd} - b_{ibd,0})^2 + \sum_{iba} K_{\theta,iba} (\theta_{iba} - \theta_{iba,0})^2 \\
&+ \sum_{idih} \left\{ K_{\phi,idih}^1 [1 - \cos (\phi_{idih} - \phi_{idih,0})] + K_{\phi,idih}^3 [1 - \cos 3(\phi_{idih} - \phi_{idih,0})] \right\} \\
&+ \sum_{i<j=3}^{nat\ contact} \varepsilon_{go,ij} \left[ 5 \left( \frac{r_{ij0}}{r_{ij}} \right)^{12} - 6 \left( \frac{r_{ij0}}{r_{ij}} \right)^{10} \right] \\
&+ \sum_{i<j=3}^{non-native} \varepsilon_{ev} \left( \frac{d}{r_{ij}} \right)^{12}
\end{align*}
1. \( \mathbf{R} \): Cartesian coordinates of the simulated protein as \( 3n_{aa} \)-dimensional vector

2. \( b_{ibd} \): \( ibd \)-th virtual bond length defined as \( |\mathbf{r}_{ibd+1} - \mathbf{r}_{ibd}| \) of the simulated protein, where \( \mathbf{r}_i \) stands for the Cartesian coordinate of the \( i \)-th amino acid (=CG particle). \((1 \leq ibd \leq n_{aa} - 1)\)

3. \( \theta_{iba} \): \( iba \)-th bond angle between two consecutive virtual bond vectors, \( \mathbf{r}_{iba} - \mathbf{r}_{iba+1} \), and \( \mathbf{r}_{iba+2} - \mathbf{r}_{iba+1} \) of the simulated proteins \((1 \leq iba \leq n_{aa} - 2)\)

4. \( \phi_{idih} \): \( idih \)-th dihedral angle around the \( idih \) + 1-th virtual bond \( \mathbf{r}_{idih+2} - \mathbf{r}_{idih+1} \) of the simulated protein. \((1 \leq idih \leq n_{aa} - 3)\)

5. \( r_{ij} \): Distance between \( i \)-th and \( j \)-th amino acids of the simulated proteins.

6. \( \mathbf{R}_n \): Cartesian coordinates of the native (sometime called reference or fiducial) structure as \( 3n_{aa} \)-dimensional vector, which corresponds to the bottom of the folding funnel.

7. All other parameters that have the subscript 0 are the constants which have the values of the corresponding variables at the native structure \( \mathbf{R}_0 \).

8. \( K_{b,ibd}, K_{\theta,iba}, K^{(1)}_{\phi,idih}, K^{(3)}_{\phi,idih}, \varepsilon_{go,ij}, \varepsilon_{ex}, \) and \( d \) are the parameters. For the former 5 parameters, Clementi et al.’s original model uses homogeneous setting, i.e., the same values for the entire systems. In CafeMol, however, one can uses site-specific parameters, i.e., the parameter values that depend on residues, upon request.

9. \( \sum_{i<j-3}^{nat contact} \): Summation over “native contact pairs”, which are pairs of amino acids that are physically close to each other at the native (or the reference) structure. If one of the non-hydrogen atoms in the \( i \)-th amino acid is within a threshold distance (inside CafeMol, defined by \( dfcontact \)) from one of non-hydrogen atoms in the \( j \)-th amino acid, we define the pair of the \( i \)-th and the \( j \)-th amino acids as being native contact. Only non-local pairs defined by \( i < j - 3 \) are taken into account.

Here, we emphasize that, even though CafeMol uses one bead (most often located at \( \alpha \) atom) per amino acid as the dynamic variable in MD simulations, the native contacts are defined by using ALL-ATOM INFORMATION of the native (reference) structure. Thus, the pdb structure given for the native structure must contain all-atom coordinates.

10. \( \sum_{i<j-3}^{non-native} \): Summation over pairs that are not in the native contact pair set. Only non-local pairs defined by \( i < j - 3 \) are taken into account.

Inside CafeMol, the parameter values of \( K_{b,ibd}, K_{\theta,iba}, K^{(1)}_{\phi,idih}, K^{(3)}_{\phi,idih}, \varepsilon_{go,ij} \) are expressed as the products of several factors.

For example, \( K_{b,ibd} \), which has the internal name “\( \text{coef}_{bd} \)” (\( \text{kcal/mol}/\text{Å}^2 \)) is expressed as

\[
\text{coef}_{bd} = \text{factor}_{bd} \times \text{correct}_\text{mgo} \times \text{cbd} \times \text{energy}_\text{unit}_\text{protein} \tag{3.2}
\]

where “\( \text{factor}_{bd} \)” and “\( \text{correct}_\text{mgo} \)” are \textit{unity, by default}, and the value of “\( \text{cbd} \)” is indicated in “./para/protein,para” file, and one can change it in the input file, as is described in Chapter 5. The “\( \text{energy}_\text{unit}_\text{protein} \)” is a scaling factor that (approximately) connects energy scale of the protein model to kcal/mol unit. These data will be output in the native-info file (See Chapter 4).

The form is essentially the same for all the local terms. \( K_{\theta,iba}, K^{(1)}_{\phi,idih}, K^{(3)}_{\phi,idih} \), respectively, are termed “\( \text{coef}_{ba} \)” , “\( \text{coef}_{dih1} \)” , and “\( \text{coef}_{dih2} \)”, which are given as

\[
\text{coef}_{ba} = \text{factor}_{ba} \times \text{correct}_\text{mgo} \times \text{cba} \times \text{energy}_\text{unit}_\text{protein} \tag{3.3}
\]

\[
\text{coef}_{dih1} = \text{factor}_{dih} \times \text{correct}_\text{mgo} \times \text{cdih},1 \times \text{energy}_\text{unit}_\text{protein} \tag{3.4}
\]

\[
\text{coef}_{dih3} = \text{factor}_{dih} \times \text{correct}_\text{mgo} \times \text{cdih},3 \times \text{energy}_\text{unit}_\text{protein} \tag{3.5}
\]
3.2. MODEL AND ENERGY FUNCTION

Here, “factor XXX” are all unity, by default, “correct_mgo” is related to multiple Go model described later, but for now it should be unity. The values of cba, cdih\_1, and cdih\_3 are given in “./para/protein.para” file.

For the contact energy, $\varepsilon_{go,ij}$, we express “coef\_go”(kcal/mol) as

$$coef\_go = \text{factor\_go} \times \text{icon\_dummy\_mgo} \times cdih\_1 \times \text{energy\_unit\_protein} \quad (3.6)$$

Here, “factor\_go” is, by default, unity. The parameter “icon\_dummy\_mgo” is unity for most pairs, but when the pair is “dummy contact” (see above), then “icon\_dummy\_mgo” should be zero. “cdih\_1” is defined in the file “./para/protein.para” file.

For some tailored simulations, one can provide non-unity “factor XXX” to introduce site-specific parameters. See Chapter 5 for how to do it.

### 3.2.2 Atomic interaction based CG (AICG) model for proteins (AICG1)

In the above Go model, the parameters $K_{\phi,ibd}$, $K_{\theta,ibd}$, $K_{\phi,iddh}$, $K_{\theta,iddh}$, and $\varepsilon_{go,ij}$ are independent of the secondary structures and of the interacting residues, which implies that all the information coded by amino acid sequence of a protein is represented by the protein native structure. For better chemical specificity, one may want to use sequence-dependent parameters. Indeed, it was found that in some cases, such chemical specificity can be crucial for the protein folding and other functional dynamics. Undoubtedly, appropriately implementing the interaction specificity into the Go model can improve the description of the protein dynamics. In CafeMol, based on the work of Li, Wolynes, & Takada 2011(Li et al., 2011), we provide a way to implement such kind of Go model with sequence-specific interactions, of which parameters were determined based on all-atom AMBER energy with an implicit solvent model by using a multiscale approach. This new Go model is called as atomic interaction based CG (AICG1) model.

In the AICG1 model, the interaction strength between the natively interacting residues $i$ and $j$ is written as

$$\varepsilon_{go,ij} = \varepsilon_{go}w_{go,ij}$$

where $\varepsilon_{go}$ and $w_{go,ij}$ being the relative weight of the interactions for each pair of contacting residues and the average of the nonlocal native interactions, respectively. The $w_{go,ij}$ controls the heterogeneity of the nonlocal native interactions. In a simple version of AICG1 model, CafeMol provides an automatic estimate of $w_{go,ij}$ via a linear regression to AMBER energy in terms of some atomic details of the residue contacts (See the supporting information to Li et al 2011). Alternatively, CafeMol allows the user to provide a parameter file containing all the interaction strengths pre-calculated by the user.

In the AICG model, the residues with different secondary structures have different bond angle and dihedral angle interaction parameters. The residues are assigned to one of the four major secondary structures, i.e., $\alpha$-helix, $\beta$-strand, turn, and random coil, based on the DSSP method(Kabsch and Sander, 1983). We also assign an independent parameter for the bond angles and dihedral angels which contain glycine. The nonlocal parameter $\varepsilon_{go}$ and all the local parameters are generic, and optimized by using the fluctuation matching method. In CafeMol, these optimized parameters are contained in the parameter file.

Currently, the AICG1 model is implemented only for proteins.

### 3.2.3 AICG with the flexible local potential (AICG2)

The energy function used in the Go model and the above AICG1 model formulates a rather stiff local potential, especially for the bond angle term which uses a harmonic potential. Based on the work of Li et al, 2012(Li et al., 2012), CafeMol provides the options to use a different energy function with a flexible local potential, which is given by
\begin{equation}
V_{AICG2}(R|R_0) = \sum_{ibd} K_{ibd}(b_{ibd} - b_{ibd,0})^2 + V_{loc}^{flip} + \sum_{i+2 \leq j \leq i+3} \varepsilon_{loc,ij} \exp \left( \frac{-(r_{ij} - r_{ij,0})^2}{2W^2_{ij}} \right) \\
+ \sum_{i<j-3} \varepsilon_{go,ij} \left[ 5 \left( \frac{r_{ij0}}{r_{ij}} \right)^{12} - 6 \left( \frac{r_{ij0}}{r_{ij}} \right)^{10} \right] \\
+ \sum_{i<j-3} \varepsilon_{ev} \left( \frac{d}{r_{ij}} \right)^{12}.
\end{equation}

In the above energy function, $V_{loc}^{flip}$ is a generic flexible local potential for the virtual bond angles and dihedral angles, and was constructed by analyzing loop structures in protein structure database (See the next subsecion on the flexible local potential. Specifically, $V_{loc}^{flip} = \sum V_{stat} + \sum V_{state}$ ). It considers the propensities of secondary structures, therefore can reasonably describe the conformational distribution of the unfolded states or intrinsically disordered proteins/regions. More detailed introduction to the above generic local potential is given in the next sub-section. The third term is the structure-based local contact potential that describes specific local interactions of the given protein structure. It represents the local contributions to the funnel-shaped energy landscape. The $\varepsilon_{loc,ij}$ and $W_{ij}$ in the Gaussian function represent the strength and width of the local interactions between residues $i$ and $j$. The other terms are the same as those used in the AICG model except that the parameters are re-optimized. The AICG with the above energy function is termed as AICG2. For convenience, The AICG model introduced in the previous sub-section is also termed as AICG1 model.

In the AICG2 model, the interaction strengths for the local and nonlocal terms are written as $\varepsilon_{loc,ij} = \varepsilon_{loc,w_{loc,ij}}$ and $\varepsilon_{go,ij} = \varepsilon_{go,w_{go,ij}}$, respectively. Here, the generic parameter $\varepsilon_{loc}$ represents the average strength of the local (nonlocal) interactions, and was optimized by using the fluctuation matching method. The $w_{loc,ij}(w_{go,ij})$ is the relative weight of the local (nonlocal) interactions. Similar to the AICG model, CafeMol provides an automatic estimate of $w_{loc,ij}(w_{go,ij})$. Alternatively, CafeMol allows the user to provide a parameter file containing all the local and nonlocal interaction strengths pre-calculated by the user. See Li et al Li et al. (2012) for more details.

3.2.4 Flexible local potential

In the simple off-lattice Go model described in subsection 3.2.1, virtual bond angles and virtual dihedral angles are rather stiffly restrained to the angle in the native structure. When one wants to perform simulations of protein regions which do not have a well folded structure, such as a disordered termini and a flexible linker regions, a fancy alternative way implemented in CafeMol is to use statistical local potentials for virtual dihedral angles and virtual bond angles in the disordered region as well as within the AICG2 model. Indeed, it was found that such a statistical approach reproduced the profiles of small angle X-ray scattering and NMR residual dipolar coupling of the intrinsically disordered region reasonably well (Terakawa and Takada, 2011).

In order to construct the statistical potential energy functions for virtual bond angle ($V_{bas}$) and virtual dihedral angle ($V_{dih}$), we first constructed a generic set of probability distributions from the dataset of 13598 protein structures in Protein Data Bank (http://www.rcsb.org/). These structures have mutual sequence identity less than 30%. For each of the proteins, using DSSP (Kabsch and Sander, 1983) for assigning the secondary structure, we extracted four consecutive loop residues (residues which are not assigned to helix or strand). The virtual bond angles $\theta$ were classified by the amino acid type of the central residue. For every central residue types, we obtained histograms with the bin size of 10 degrees. Thus, totally, 20 probability distributions $P(\theta)$ were constructed. The virtual dihedral angles $\phi$ were classified by the central pair of amino acid types. For every pairs we obtained histograms with the bin size of 10 degrees. Thus, totally 400 probability distributions $P(\phi)$ were obtained. By comparing these 400 probability distributions, we found that the distributions are similar when the central pairs of amino acid are $R_1$:Gly, $R_2$:Pro, Ala-$R_3$, Arg-$R_3$, Asn-$R_3$, Asp-$R_3$, Cys-$R_3$, Gln-$R_3$, Glu-$R_3$, Gly-$R_3$, His-$R_3$, Ile-$R_3$, Leu-$R_3$, Lys-$R_3$, Met-$R_3$, Phe-$R_3$, Pro-$R_3$, Ser-$R_3$, Thr-$R_3$, Trp-$R_3$, Tyr-$R_3$, or Val-$R_3$ where $R_1$ represents all amino
acids, $R_2$ represents all amino acids except Gly, and $R_3$ represents all amino acids except Gly and Pro. The distribution of Gly-Pro is not similar to any other distributions, and thus is treated as an independent type. Then, we averaged the similar distributions and re-constructed totally 23 probability distribution $P(\phi)$. (Gly-Pro, and 22 types defined by the above described pairs). Then, we inverted these probability distributions in ordered to obtain the potential energy function for a virtual bond angle,

$$V_{\text{state}}^{ba} = -k_B T \ln \frac{P(\theta)}{\sin(\theta)}$$

and for a virtual dihedral angle,

$$V_{\text{state}}^{dih} = -k_B T \ln P(\phi)$$

where $k_B$ is the Boltzmann constant and $T = 300.0$ is temperature.

When we calculate a force and/or an energy for a virtual bond angle in simulations, we interpolate these tabulated values with the cubic spline interpolation to obtain a continuous and differentiable potential energy function. Outside the range of sufficient samples in the above PDB library, we employ a linear potential to expel these range of bond-angles.

When we calculate a force for a virtual dihedral angle in simulation, to satisfy the periodic boundary condition of $\phi$, we fit the tabulated data by the truncated Fourier series as

$$f(\phi) = \sum_{m} k_m \sin(m\phi) + \sum_{n} k_n \cos(n\phi) + C$$

where $k_m, k_n$, and $C$ are Fourier coefficient.

### 3.2.5 Multiple-basin model for proteins

*CafeMol* employs the multiple-basin potential (MBP) of Okazaki et al.

The MBP energy function is defined by the (smaller) eigenvalue of the characteristic equation,

$$\begin{pmatrix} V_{Go+}(R|\nu) & \Delta \\ \Delta & V_{Go+}(R|\nu) + \Delta V \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = V_{MB} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

where $V_{Go+}(R|\nu)$ is essentially the Clementi’s off-lattice Go potential $V_{Go+}(R|\nu)$ but is modified in two respects as described later. The condition that a nontrivial solution exists leads to the secular equation,

$$\det \begin{pmatrix} V_{Go+}(R_1|\nu) & V_{MB} \\ \Delta & V_{Go+}(R_2|\nu) + \Delta V - V_{MB} \end{pmatrix} = 0$$

of which the explicit solution can be written down as

$$V_{MB}(R_1,R_2) = \frac{1}{2} \left[ V_{Go+}(R_1|\nu) + V_{Go+}(R_2|\nu) + \Delta V \right] - \sqrt{\left( \frac{V_{Go+}(R_1|\nu) - V_{Go+}(R_2|\nu) - \Delta V}{2} \right)^2 + \Delta^2}$$

where

1. $\Delta$ is a coupling constant, which smoothed the connection between two Go models. The larger $\Delta$ leads to smaller barrier height between two basins.
2. $\Delta V$ is to modulate the relative energies of the two basins.
As a reaction coordinate,
\[ \chi = \ln \left( \frac{c_2}{c_1} \right) \]  
(3.11)

is a convenient quantity. It is negative when the system is in the state 1 and is positive when the system is in the state 2.

Then, we discuss about the general n-basin case. To obtain an n-basin potential, the secular equation to be solved becomes

\[
\det \begin{pmatrix}
V_{G_0+}(R|R_1) - V_{MB} & \Delta_{12} & \cdots \\
\Delta_{21} & V_{G_0+}(R|R_2) + \Delta V_2 - V_{MB} & \cdots \\
\vdots & \vdots & \ddots & \ddots \\
\Delta_{n1} & \Delta_{1n} & \cdots & \Delta_{n(n-1)} & V_{G_0+}(R|R_n) + \Delta V_n - V_{MB}
\end{pmatrix} = 0
\]  
(3.12)

where \( \Delta_{\mu\nu} = \Delta_{\nu\mu} \) is the coupling constant between basin \( \mu \) and \( \nu \), \( \Delta V_\nu \) is the relative stability of basin \( \nu \), and \( \nu > \mu = 1, 2, \ldots, n-1 \). In the current version, \textit{CafeMol} can treat up to 3-basin cases analytically. Note that for the case of \( n \geq 3 \), there is no unique definition of \( \chi \). So, in this situation, other reaction coordinate should be considered.

Now, \( V_{G_0+}(R|R_\nu) \) is to be defined. As was noted, this is conceptually just \( V_{G_0}(R|R_\nu) \) of Clementi et al. Purely by technical reasons, we need to introduce two modifications. We write the \( V_{G_0+}(R|R_\nu) \) in the sum of three terms,

\[
V_{G_0+}(R|R_\nu) = V_{local}(R|R_\nu) + V_{native-attr}(R|R_\nu) + V_{repul}(R|R_\nu)
\]  
(3.13)

where the first term is

\[
V_{local}(R|R_\nu) = \sum_{i,bd} K_{bi,ibd}(b_{ibd} - b_{i,\nu,\nu})^2 + \sum_{i,ba} K_{\theta i,iba}(\theta_{iba} - \theta_{iba,\nu})^2
\]

\[
+ \sum_{i, idih} \left[ K^{(1)}_{\phi i, idih} [1 - \cos(\phi_{idih} - \phi_{idih,\nu})] + K^{(3)}_{\phi i, idih} [1 - \cos 3(\phi_{idih} - \phi_{idih,\nu})] \right]
\]  
(3.14)

This local potential is slightly different from that of \( V_{G_0}(R|R_\nu) \): Namely, three of the \( K \)'s are now dependent on \( i \) in the following ways;

\[
K_{bi} = K_b \times \min \left[ 1, \frac{\epsilon_{b,\max}}{K_b (b_{i,1} - b_{i,2})^2} \right]
\]  
(3.15)

\[
K_{\theta i} = K_\theta \times \min \left[ 1, \frac{\epsilon_{\theta,\max}}{K_\theta (\theta_{i,1} - \theta_{i,2})^2} \right]
\]  
(3.16)

\[
K^{(1)}_{\phi i} = K^{(1)}_{\phi} \times \min \left[ 1, \frac{\epsilon_{\phi,\max}}{K^{(1)}_{\phi} [1 - \cos(\phi_{i,1} - \phi_{i,2})] + K^{(3)}_{\phi} [1 - \cos 3(\phi_{i,1} - \phi_{i,2})]} \right]
\]  
(3.17)

\[
K^{(3)}_{\phi i} = K^{(3)}_{\phi} \times \min \left[ 1, \frac{\epsilon_{\phi,\max}}{K^{(1)}_{\phi} [1 - \cos(\phi_{i,1} - \phi_{i,2})] + K^{(3)}_{\phi} [1 - \cos 3(\phi_{i,1} - \phi_{i,2})]} \right]
\]  
(3.18)

This is introduced so that spring constants should be weakened where large change is observed between two reference structures. \( \epsilon_{b,\max}, \epsilon_{\theta,\max} \) and \( \epsilon_{\phi,\max} \) are the cutoffs that define the "large change". We note that because of this, \( V_{local}(R\{R_\nu\}) \) is not just a function of the reference structure \( R_\nu \), but also depends on other reference structures.
Non-local potentials are divided into attractive-terms \( V_{\text{native-attr}} \) and repulsive terms \( V_{\text{repul}} \), and the former is given as
\[
V_{\text{native-attr}}(\mathbf{R}|\mathbf{R}_\nu) = \epsilon_{\text{go}} \sum_{i<j<3} \min_{\nu} \left[ 1, 5 \left( \frac{r_{ij,\nu}}{r_{ij}} \right)^{12} - 6 \left( \frac{r_{ij,\nu}}{r_{ij}} \right)^{10} + 1 \right]
\]
(3.19)
where the summation is over the native contact pairs in the same way as that of single Go model. The repulsive part \( V_{\text{repul}} \) is further divided into two terms \( V_{\text{repul-1}}(\mathbf{R}|\mathbf{R}_\nu) \) and \( V_{\text{repul-2}}(\mathbf{R}) \),
\[
V_{\text{repul}}(\mathbf{R}|\mathbf{R}_\nu) = V_{\text{repul-1}}(\mathbf{R}|\mathbf{R}_\nu) + V_{\text{repul-2}}(\mathbf{R})
\]
(3.20)
where
\[
V_{\text{repul-1}}(\mathbf{R}|\mathbf{R}_\nu) = \epsilon_{\text{go}} \sum_{i<j<3} \max_{\nu} \left[ 0, 5 \left( \frac{r_{ij,\nu}}{r_{ij}} \right)^{12} - 6 \left( \frac{r_{ij,\nu}}{r_{ij}} \right)^{10} \right]
\]
(3.21)
and
\[
V_{\text{repul-2}}(\mathbf{R}) = \epsilon_{\text{ev}} \sum_{i<j<3} \left( \frac{d}{r_{ij}} \right)^{12}
\]
(3.22)
where
\[
\frac{r_{ij,\nu}^{\min}}{r_{ij,\nu}^{\max}} = \frac{\min_{\nu} r_{ij,\nu}}{\max_{\nu} r_{ij,\nu}}
\]
(3.23)
Here, the repulsive-1 term is used for the pairs for which the native contact is formed at least in one of the reference structures (“type 1” and “type 2” pairs in the original paper of Okazaki et al.). We termed it as “native-related pairs”. For a particular pair \( ij \), if the pair is in native contact in the state \( \nu \), this is a (true) native contact for this state. If the pair is not in the native contact in the state \( \nu \), but it makes contact in other state, we call this pair in the “dummy contact” in the state \( \nu \). For the state \( \nu \), the true native contact set plus the dummy contact set are equal to the “native-related pairs”. The repulsive-2 term is for the pairs for which no native contact is formed in any of the reference structures (“type 3” pairs in the original paper). \( V_{\text{repul-2}}(\mathbf{R}) \) is the same form as that of Clementi et al.’s Go model.

We note that the attractive part has the same shape as that of Clementi et al., but the repulsive part of the LJ-like potential is modified for the native-related pairs so that the repulsive part is common in all the states \( \nu \). The \( r_{ij,\nu}^{\min} \) is the smallest distance between \( i \) and \( j \) among all the states \( \nu \) for which this pair \( ij \) is in native contact. We note that because of this, \( V_{\text{repul-1}}(\mathbf{R}|\mathbf{R}_\nu) \) is not just a function of the reference structure \( \mathbf{R}_\nu \), but also depends on other reference structures.

Inside CafeMol, the softening correction such as \( K_{\theta_1} = K_{\theta} \times \min \left[ 1, \frac{r_{\theta,\max}}{r_{\theta,\nu}^{\max}} \right] \) is reflected by the factor “correct_mgo” in
\[
\text{coef}_{ba} = \text{factor}_{ba} \times \text{correct}_\text{mgo} \times \text{cbe} \times \text{energy_unit} \times \text{protein}
\]
(3.24)
where “correct_mgo” is mostly unity, but for the bond angles that change quite largely “correct_mgo” value smaller than unity is used. This correction will be done automatically by default.

Currently, multiple basin (Go) model with AICG2 is not supported.

### 3.2.6 RNA model

The RNA model implemented in CafeMol is conceptually similar to the off-lattice Go model for proteins (3.2.1). (Hori and Takada, 2012) In the model, RNA is coarse-grained using three particles, phosphate(P), sugar(S) and base(B), for each nucleotide. This scale, sometimes called three-site-per-nucleotide, has been used in some other coarse-graining studies of both RNA (Hyeon and Thirumalai, 2005) and DNA (Knotts et al., 2007). The scale is thought to be a reasonable partition of molecule in terms of chemical property. It is, additionally, roughly consistent with coarse-graining scale of the protein model in CafeMol, i.e., one-beads-per-residue. The CG particles of phosphates are placed at positions of phosphorus atom of each phosphate group, CG particles of sugars are placed at the center of sugar-ring atoms (User also can select the center of whole-sugar atoms, or C4’ atom position), and CG particles of bases are placed at positions of N1 atom for purine-base and N3 atom for pyrimidine-base (User also can select the center of mass of base moiety).

Primary target of this model is functional RNAs and their complexes with proteins which have unique native structures in solution, such as tRNA and ribosome. As in the case of proteins, we assume that the fluctuation
around the native structure is, in a good approximation, determined by its structural topology of its native state, and thus the structure-based potential works well. The potential is described by several terms as in the case of proteins,

\[ V_{RNA} = V_{local} + V_{contact} + V_{excluded} \]  

(3.25)

The functional form of the local potential is the same as the Clementi-type Go model for proteins. From statistical analysis of known structures in PDB, we found it is enough to capture the characteristic distribution that base is treated as two different types, purine(R) and pyrimidine(Y). Due to existence of four different types of particle, P, S, R and Y, we assign different coefficients for each type of local virtual-bond lengths and virtual bond angles. Although, for bond angles and dihedral angles, there are four possible ways to choose a set of particles which define these angles, we use only three of them. Concretely, we do not include bond angles made by B(i)-S(i)-P(i+1) and dihedral angle by B(i)-S(i)-P(i+1)-S(i+1), where i is a residue number. Then, we have,

\[
V_{local} = \sum_{\eta(bd) \in PS, SP, SR, SY} K_{b}^{\eta}(r_{ibd} - r_{0}^{\eta})^{2}
\]

\[
+ \sum_{\eta(iba) \in PSP, SPS, SPR, SPY} K_{\theta}^{\eta}(\theta_{iba} - \theta_{0}^{\eta})^{2}
\]

\[
+ \sum_{\eta(idih) \in PPS, PSR, SPS, SY} \{K_{\phi}^{\eta}(1 - \cos(\phi_{idih} - \phi_{0}^{\eta})) + \frac{1}{2}K_{\phi}^{\eta}(1 - \cos3(\phi_{idih} - \phi_{0}^{\eta}))\}
\]

where \( \eta \) represents one of PS, SP, SR or SY in the first term (bond-length term), PSP, SPS, SPR, or SPY in the second term (bond-angle term), and PPS, PSR, SPS, or PSY in the third term (dihedral-angle term).

Non-local interactions described by structure-based potential are also important for RNAs similarly to proteins to maintain their tertiary structures. We use the same form as that of Clementi et al for proteins. Namely,

\[
V_{contact} = \sum_{i,j \in S,B} \varepsilon_{\xi} \left[ 5 \left( \frac{r_{0ij}}{r_{ij}} \right)^{12} - 6 \left( \frac{r_{0ij}}{r_{ij}} \right)^{10} \right]
\]

(3.27)

In contrast with proteins, nucleic acids have characteristic interaction which play central role to form typical single or double stranded helix. We treat such interactions, base pair and base-stacking, distinctively from general contact interaction by using different coefficients. In our model, two residues make base pair interaction if two or more hydrogen bond are formed between them in the reference structure. The interaction type is further discriminated by the number of hydrogen bonds; two (\( \varepsilon_{\xi} = \varepsilon_{BP2} \)), three, or more (\( \varepsilon_{\xi} = \varepsilon_{BP3} \)) because the interaction strength would depend on the number of formed hydrogen bonds. Base stacking is another important feature (\( \varepsilon_{\xi} = \varepsilon_{ST} \)). To exclude flipped base from this interaction, we define base stacking by following two conditions which derived from preliminary statistical survey (SI). (1) The distance of the nearest neighbor atoms of them is less than 6.0 angstrom in the native structure. (2) The virtual dihedral angle formed by B(i)-S(i)-S(i+1)-B(i+1) is smaller than 40° in the native structure. All other contacts formed by base and sugar particles are detected as “contact” interaction (\( \varepsilon_{\xi} = \varepsilon_{con} \) where \( \eta \) is one of S-S, S-B or B-B). The detection is atomic-based as in the case of proteins, and the cutoff distance is 5.5 angstrom.

Particles which has no non-bonded interactions explained above have only excluded-volume effect.

\[
V_{excluded} = \sum_{i,j \text{ s.t. } non - local} \varepsilon_{ex} \left( \frac{d}{r_{ij}} \right)^{12}
\]

(3.28)
3.2. MODEL AND ENERGY FUNCTION

Here, CG particles i and j are defined as non-local if they cannot be connected through 3 consecutive virtual bonds.

We can employ Go-like potential to include complex structure of RNA and protein as well.

$$V_{\text{complex}} = \sum_{i \in \text{pro}} \sum_{j \in S, B} \varepsilon_{\eta} \left[ 5 \left( \frac{r_{ij}^0}{r_{ij}} \right)^{12} - 6 \left( \frac{r_{ij}^0}{r_{ij}} \right)^{10} \right]$$ (3.29)

where $\eta$ is either $\text{pro} - S$ for interaction between protein and sugar, or $\text{pro} - B$ for that between protein and base.

3.2.7 DNA model

For DNA, CafeMol implements a CG model (designated as 3SPN.1 model) developed by de Pablo group (Knotts et al., 2007; Sambriski and de Pablo, 2009). In this model, CG particles represent the three chemical moieties comprising a nucleotide, sugar, phosphate, and nitrogenous base. The CG particles for sugars and for phosphates are placed at the center of mass of sugars and phosphates, respectively. For purine bases (adenine and guanine), the CG particles are located at the N1 position, whereas for pyrimidine bases (cytosine and thymine), the CG particles are located at the N3 positions. This CG potential energy is made of a balanced mixture of structure-based terms targeting for B-type double-stranded (ds) DNA and physico-chemical interactions.

The 3SPN.1 model is defined by the potential energy function,

$$V_{\text{DNA}} = V_{\text{bond}} + V_{\text{angle}} + V_{\text{dihedral}} + V_{\text{stuck}} + V_{\text{base}} + V_{\text{excluded}} + V_{\text{solv}} + V_{\text{ele}}.$$ 

$V_{\text{bond}}$ is the potential energy term for virtual bond stretching,

$$V_{\text{bond}} = \sum_i k_{\text{DNA}} \left( \frac{r^i - r^i_0}{r^i} \right)^2 + k_{\text{DNA}} \left( \frac{r^i - r^j_0}{r^i} \right)^4,$$

where $k_{\text{DNA}} = 0.1839$ and $k_{\text{DNA}} = 183.9$ are the same constants for all the bonds, $r^i$ is the i-th virtual bond length, and $r^i_0$ is the length in the canonical B-type dsDNA structure. $V_{\text{angle}}$ is the potential energy term for virtual bond angle bending,

$$V_{\text{angle}} = \sum_i k_{\text{DNA}} \left( \theta^i - \theta^i_0 \right)^2,$$

where $k_{\text{DNA}} = 128.73$ is constant, $\theta^i$ is the i-th virtual bond angle, and $\theta^i_0$ is the bond angle in the canonical B-type dsDNA structure. $V_{\text{dihedral}}$ is the potential energy function for the dihedral angle twisting,

$$V_{\text{dihedral}} = \sum_i k_{\text{DNA}} \left[ 1 - \cos \left( \phi^i - \phi^i_0 \right) \right],$$

where $k_{\text{DNA}} = 5.1492$ is constant, $\phi^i$ is the i-th virtual dihedral angle, and $\phi^i_0$ is the dihedral angle in the canonical B-type dsDNA structure.

$V_{\text{stuck}}$ is the potential energy function for base stacking,

$$V_{\text{stuck}} = \sum_{i<j} 4 \varepsilon \left[ \left( \frac{\sigma_{ij}^\text{DNA}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}^\text{DNA}}{r_{ij}} \right)^{10} \right],$$

where $\varepsilon = 0.1839$ is constant, $\sigma_{ij}^\text{DNA}$ is pair-dependent constant (based on the dsDNA structure) and $r_{ij}$ is the distance between i-th and j-th beads. The summation is taken only for the bead pairs that are in a
single strand of DNA and the distance between them is within 9.0 Å in the native structure. $V_{\text{base}}^{\text{DNA}}$ is the potential energy function for Watson-Crick type base pairing,

$$V_{\text{base}}^{\text{DNA}} = \sum_{i<j} 4 \varepsilon_{\text{bij}}^{\text{DNA}} \left[ 5 \left( \frac{\sigma_{\text{bij}}^{\text{DNA}}}{r_{ij}} \right)^{12} - 6 \left( \frac{\sigma_{\text{bij}}^{\text{DNA}}}{r_{ij}} \right)^{10} \right],$$

where $\varepsilon_{\text{bij}}^{\text{DNA}}$ and $\sigma_{\text{bij}}^{\text{DNA}}$ are pair-dependent constant. $\varepsilon_{\text{bij}}^{\text{DNA}}$ is 0.3678 for AT pair and 0.4656 for CG pair. $\sigma_{\text{bij}}^{\text{DNA}}$ is 2.9002 for AT pair, and 2.8694 for CG pair (based on the dsDNA structure). The summation is taken only for all complementary (i.e., Watson-Crick type) base pairs that do not participate in $V_{\text{DNA}}^{\text{stuck}}$.

A complementary base pair is considered to be hydrogen-bonded when the separation between bases is $r_{ij} < (\sigma_{\text{bij}}^{\text{DNA}} + 2.0$ Å) in the native structure. $V_{\text{excluded}}^{\text{DNA}}$ is the potential energy function for excluded volume effect,

$$V_{\text{excluded}}^{\text{DNA}} = \sum_{i<j} \begin{cases} 4 \varepsilon \left( \frac{\sigma_{\text{bij}}^{\text{DNA}}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{\text{bij}}^{\text{DNA}}}{r_{ij}} \right)^{6} & \text{if } r_{ij} < r_{\text{coff}} \\ 0 & \text{if } r_{ij} \geq r_{\text{coff}} \end{cases},$$

where $\sigma_{\text{bij}}^{\text{DNA}} = 2^{-1/6} r_{\text{coff}}$ and $r_{\text{coff}}$ is constant. $r_{\text{coff}}$ is 1.00 Å for mismatch pairs and 6.86 Å otherwise. The summation is taken only for natively non-contacting pairs. $V_{\text{solv}}^{\text{DNA}}$ is the potential energy function for solvation energy,

$$V_{\text{solv}}^{\text{DNA}} = \sum_{i<j} E_{\text{s}}^{\text{DNA}} \left[ 1 - e^{-\alpha (r_{ij} - r_{s}^{\text{DNA}})} \right]^2,$$

where $E_{\text{s}}^{\text{DNA}}$ is an ion concentration dependent constant, $\alpha = 5.333 \text{Å}^{-1}$ and $r_{s}^{\text{DNA}} = 13.38$ Å are constant. The summation is taken only for inter-strand sugar bead pairs. $V_{\text{ele}}$ is the same as that described in “Electrostatic interaction” section and is applied between phosphate pairs except those involved in $V_{\text{bond}}^{\text{DNA}}$.

This model was shown to capture the ionic strength dependency of the melting temperature, the persistence length, and the heat capacity profiles of dsDNA.

3.2.8 Elastic network model of proteins and RNAs

*CafeMol* has a simple Tirion’s elastic network model (ENM) defined by

$$V_{\text{enm}} = K \sum_{i<j}^{\text{n nat contact}} (r_{ij} - r_{ij,0})^2 \quad (3.30)$$

where the definition of the native contact is the same as that of the Go model. One can use the ENM for proteins and RNAs.

Currently, the ENM in *CafeMol* is quite limited. In particular, one cannot mix the ENM with other models within one simulation system.

3.2.9 Electrostatic interaction

*CafeMol* implements the Debye-Huckel type electrostatic interactions. In MKSA unit, the electrostatic interaction is expressed as

$$V_{\text{ele}} = \sum_{i<j}^{N} \frac{q_i q_j}{4 \pi \varepsilon_0 \varepsilon_r r_{ij}} e^{-r_{ij}/\kappa_D} \quad (3.31)$$

$$\kappa_D = \left( \frac{\varepsilon_0 \varepsilon_r k_B T}{2 N_A e^2 I} \right)^{0.5} \quad (3.32)$$
where $q_i$ is charge, $\epsilon_0$ is electric constant, $\epsilon_k$ is (dimensionless) dielectric constant, $\kappa_D$ is the so-called Debye length, $N_A$ is Avogadro’s number, and $I$ is the ionic strength. The ionic strength is defined as $I = 0.5 \sum z_i^2 c_i$ where $z_i = q_i/e$ and the $c_i$ is the molar density per m$^3$. Once we changed the unit to the CafeMol unit and use molar density per liter for $c_i$, the formula are transferred to

$$V_{\text{ele}} = 332 \sum_{i<j}^N \frac{q_i q_j}{\epsilon_k r_{ij}} e^{-r_{ij}/\kappa_D}$$

where the ion strength $I = 0.5 \sum z_i^2 c_i$ and $c_i$ is molar per liter (M). The ion strength is 1 for 1M NaCl. If we use a standard dielectric constant for water 78 and the temperature 300K, the Debye length for 1M NaCl solution is about 3.04 Å, and that for 10mM NaCl solution is 30.4 Å.

This potential is optional. When it is used, one needs to specify some parameters in the input file. By the default use, in proteins, all the Lys, Arg, and His residues have +1 charges, and all the Asp and Glu residues have -1 charges although one can change the assignment in more specific way.

### 3.2.10 Hydrophobic interaction

The hydrophobic (HP) interaction is modeled by a many-body energy function, which has a similar functional form to that was used in Fujitsuka, et al. (Fujitsuka et al., 2004);

$$V_{\text{HP}} = -c_{\text{HP}} \sum_{i\in\text{HP}} \epsilon_{\text{HP},A(i)} S_{\text{HP}}(\rho_i)$$

(3.33)

The “buriedness” of the particle $i$ is defined by:

$$S_{\text{HP}}(\rho) = \begin{cases} 1, & \rho \geq 1 \\ c_{\text{linear}} \rho + 0.5(1 - c_{\text{linear}}) \left[ 1 + \cos \left( \frac{\pi(1 - \rho)}{\rho - \rho_{\text{min}}} \right) \right], & \rho_{\text{min}} < \rho < 1 \\ 0, & \rho \leq \rho_{\text{min}} \end{cases}$$

(3.34)

where the local density $\rho_i$ for particle $i$ is calculated by:

$$\rho_i = \frac{\sum_{j\in\text{HP},j\neq i} n_{A(j)} u_{\text{HP}}(r_{ij}, r_{\text{min}}, A(i), A(j), r_{\text{max}}, A(i), A(j))}{n_{\text{max},A(i)}}$$

(3.35)

The function $u_{\text{HP}}$ represents the degree of the contact between particles $i$ and $j$ and is defined as:

$$u_{\text{HP}}(r, r_{\text{min}}, r_{\text{max}}) = \begin{cases} 1, & r \leq r_{\text{min}} \\ 0.5 \left( 1 + \cos \frac{\pi(r - r_{\text{min}})}{r_{\text{max}} - r_{\text{min}}} \right), & r_{\text{min}} < r < r_{\text{max}} \\ 0, & r \geq r_{\text{max}} \end{cases}$$

(3.36)

Some variables and parameters are explained below:

1. $c_{\text{HP}}$: a parameter that scales the overall strength of the hydrophobic interactions
2. $A(i)$: the type of the particle $i$. In the current version of CafeMol, 21 types are considered: $A(i) = 1, 2, ..., 20$ means the 20 types of amino acids and $A(i) = 21$ simply represents all non-amino acid particles.
3. $\epsilon_{\text{HP},A}$: particle-type-specific parameter that reflects the hydrophobicity of particles. The values are set 0 for non-hydrophobic amino acids.
4. $c_{\text{linear}}$: the constant for the calculation of particle buriedness
5. $\rho_{\text{min}}$: the threshold that defines the minimal local density of a particle
6. $n_{A}$: the number of atoms that the particle type $A$ represents
7. $n_{\text{max},A}$: the maximum coordination number for particle type $A$

8. $r_{ij}$: the distance between the hydrophobic particles $i$ and $j$

9. $r_{\text{min},A,B}$: one entry from the $21 \times 21$ parameter matrix $r_{\text{min}}$; it defines the cutoff for the minimal distance between particle types $A$ and $B$

10. $r_{\text{max},A,B}$: one entry from the $21 \times 21$ parameter matrix $r_{\text{max}}$; it defines the cutoff for the maximal distance between particle types $A$ and $B$.

For all the parameters of hydrophobic interaction mentioned above, you can find their values in the parameter file (see Chapter 4.). One can limit the range of hydrophobicity by specifying it in the input file.

3.2.11 Excluded volume interaction

For any non-local pair of CG particles $i$ and $j$, one can use generic excluded volume interaction,

$$\sum_{ij} \varepsilon_{\text{ev}} \left( \frac{d}{r_{ij}} \right)^{12}$$

Here, CG particles $i$ and $j$ are defined as non-local if they cannot be connected through 3 consecutive virtual bonds.

When one uses Go model or its variants together with the excluded volume term, the on-off of excluded volume interaction for each pair is tightly coupled with the Go contact interaction. Namely, if an $ij$ pair is included in the Go contacts, the excluded volume interaction is turned off for this pair. Otherwise, the excluded volume interaction is turned on. Note that we have a cutoff $1.0 \times 10^{-6}$ of the Go contact energy strength; if the Go contact interaction coefficient between the pair is weaker than this cutoff, it is regarded as no Go contact and thus the excluded volume interaction is turned on.

3.2.12 Implicit ligand model

Often, protein conformational change is coupled with a ligand binding, and one wants to take into account the ligand binding dynamics into the CG simulations. Explicitly including a CG ligand is one straightforward way, but there is another simpler way, too. Namely, one can take ligand binding into consideration implicitly. This was first reported in Okazaki and Takada 2008 (Okazaki and Takada, 2008). Here, we describe the method.

We describe that ligand binding in two state manner. Namely, the ligand is either bound (B) or unbound (U) to a protein. In the unbound state, the protein has just its intra-energy, represented typically by the multiple basin model, whereas the protein in the bound state has the intra-energy plus the ligand binding energy, $V_{\text{imp-lig}}$. This ligand binding energy $V_{\text{imp-lig}}$ does not contain the explicit coordinates of the ligand atoms, but is a function of the Cartesian coordinates of ligand-mediated sites of the protein in a reference state. It takes negative and large absolute value when the local environment around the binding pocket is close to that of the reference conformation.

Concretely, the $V_{\text{imp-lig}}$ can be

$$V_{\text{imp-lig}} = \sum_{\text{ligand-mediated contact-pairs}} -c_{\text{lig}} \varepsilon_{\text{go}} \exp \left[ -\frac{(r_{ij}/r_{0ij} - 1)^2}{2(\sigma/r_{0ij})^2} \right]$$

or

$$V_{\text{imp-lig}} = \sum_{\text{ligand-mediated contact-pairs}} c_{\text{lig}} \varepsilon_{\text{go}} \left[ 5 \left( \frac{r_{0ij}}{r_{ij}} \right)^{12} - 6 \left( \frac{r_{0ij}}{r_{ij}} \right)^{10} \right]$$

where the "ligand-mediated contact-pairs" means the amino acid pairs that satisfy the following three conditions. a) Both of amino acids in the pair are involved in the binding-sites, which are provided in the input file. b) The pair is not involved in the native contact. c) at least one heavy atom in one amino acid is within 10Å from at least one atom in the other amino acid.
In time propagation, the protein conformation is moved by the standard MD simulation (described below), whereas the ligand-binding state (B or U) is stochastically changed by the rates $k_b$ (binding) and $k_u$ (unbinding) implemented as the Metropolis Monte Carlo (MC) scheme. While in the unbound state, a ligand molecule reaches the binding pocket at every time with probability $p = k_b \Delta t_b$. With this probability, the state changes to the bound (B) one. Here $k_b$ is the apparent first-order rate for binding. While in the bound state, at every $\Delta t_u$ time, the bound ligand has chance to dissociate at a probability $\exp(-|V_{\text{imp}} - V_{\text{lig}}|/k_B T)$. The mixed MD-MC scheme thus described here is a convenient way of simulating protein conformational dynamics coupled with ligand binding.

### 3.2.13 Explicit ligand model

Alternatively, small ligand molecules can be modeled explicitly as a rigid linear chain, in which the sequence of ligand beads is defined in the input PDB file of ligand. The energy function for explicit ligands is expressed by:

$$V_{\text{ligand}}(\mathbf{R}_l | \mathbf{R}_0) = \sum_{ibd} K_{b,l} (b_{ibd,l} - b_{ibd,0l})^2 + \sum_{iba} K_{\theta,l} (\theta_{iba,l} - \theta_{iba,0l})^2 + \sum_{idih} K_{\phi,l} (\Delta \phi_{idih,l})^2 + \sum_{i,j} \varepsilon_{ev,l} \left( \frac{d_{ij}}{r_{ij}} \right)^{12}$$

(3.37)

where

$$\Delta \phi_{idih,l} = \left\{ \begin{array}{ll} \phi_{idih,l} - \phi_{idih,0l} + 2\pi & \text{if } \Delta \phi \leq -\pi \\ \phi_{idih,l} - \phi_{idih,0l} & \text{if } -\pi < \Delta \phi < \pi \\ \phi_{idih,l} - \phi_{idih,0l} - 2\pi & \text{if } \Delta \phi \geq \pi \end{array} \right. \quad (3.38)$$

the bond length, bond angle, and dihedral angle are defined by the same way as $V_{\text{Go}}(\mathbf{R}|\mathbf{R}_\nu)$ of Clementi et al. The subscript 0 means the variables in the reference structure. Usually, the spring constants, $K_{b,l}$, $K_{\theta,l}$, and $K_{\phi,l}$ are set as sufficiently large values (see Chapter 4). Note that for the repulsive term, $i$ and $j$ must be in the different molecules.

### 3.2.14 Lipid model (in preparation)

to be written

### 3.3 Molecular move algorithm

#### 3.3.1 Minimization

to be written

#### 3.3.2 Constant temperature “Newtonian” dynamics (Berendsen thermostat or velocity rescaling)

As a simple way to realize (an approximate) constant temperature ensemble, CafeMol employs “Newtonian” dynamics with Berendsen thermostat. Newtonian dynamics is defined by the Newton equation,

$$m_i \frac{d^2 \mathbf{r}_i}{dt^2} = \mathbf{f}_i$$

(3.39)

where $\mathbf{r}_i$ is the Cartesian coordinate of the $i$-th mass-point (CG particle), $\mathbf{f}_i$ is the force derived from the total potential energy function $V$ as $\mathbf{f}_i = -\frac{\partial V}{\partial \mathbf{r}_i}$, and $m_i$ is the mass.
For time-integration, *CafeMol* uses the so-called velocity-version of Verlet algorithm (Smit and Frenkel, 2006).

\[
\mathbf{r}_i(t + h) = \mathbf{r}_i(t) + \mathbf{v}_i(t)h + \frac{1}{2}h^2 \frac{\mathbf{f}_i(t)}{m_i}
\]  

where \( \mathbf{r}_i \) is the position, \( \mathbf{v}_i \) the velocity, and \( h \) is the (small) time step of MD simulation. Direct application of this algorithm, of course, leads to constant-energy Newtonian dynamics. To realize, instead, the constant-temperature “Newtonian” dynamics, Berendsen algorithm (Berendsen et al., 1984) scales the velocity as

\[
\mathbf{v}_i(t + h) \leftarrow s\mathbf{v}_i(t + h)
\]

where \( s \) is the scaling factor. The factor \( s \) is to uniformly modify the velocity so that the resulting velocities have kinetic energy closer to the target temperature \( T \). First, we define the instantaneous temperature \( T_i(t) \) as

\[
\frac{3}{2}Nk_B T_i(t) = \sum_{i=1}^{N} \frac{1}{2}m_i v_i(t)^2
\]

where \( N \) is the number of mass-points. With this, the scaling factor \( s \) is given as

\[
s = \sqrt{1 + \eta \left( \frac{T}{T_i(t + h)} - 1 \right)}
\]

where the parameter \( \eta \) (0 < \( \eta \) ≤ 1) determines how quickly the temperature is enforced to the target one. When \( \eta = 1 \), the method is sometimes called the velocity scaling method (Woodcock) where the velocity is scaled so that the instantaneous temperature always becomes the target temperature.

In *CafeMol*, \( h \) is called “tstep.size”, and its value has to be given in “<<<< md_information”. Typical values in Newtonian dynamics would be 0.05 ~ 0.1.

In *CafeMol*, \( \eta \) is called “velo_adjst”. Its default value is set to 1.0 (thus, we use the so-called velocity scaling method) in general.para file, and can be changed in the input block “<<<< redefine para”.

With this dynamics, while simulating protein folding, we often observed that, kinetic energy was gradually shifted from the internal degrees of freedom to the overall translational and/or rotational degrees of freedom, which resulted in extending the protein chain into nearly straight line and freezing the internal motion. This occurred because of some small non-zero angular momentum and thus it is highly recommended to prohibit overall translational and rotational motions when the Newtonian dynamics is used.

Also, we warn that Newtonian dynamics leads to quite unnatural phenomena in certain situations. For example, when two proteins dissociate each other, the relative motion between two centers of mass is the constant-velocity straight move, which is of course very strange because proteins are, in reality surrounded by fluctuating solvent molecules. Simply, when (nearly) isolated translational and rotational motions of biomolecules are part of your interest, Newtonian dynamics is not appropriate.

### 3.3.3 Constant temperature “Newtonian” dynamics (Nose-Hoover thermostat)

*CafeMol* has a Nose-Hoover (NH) chain thermostat routine (Martyna et al., 1992) that realizes the constant temperature ensemble more rigorously than the Berendsen thermostat. More specifically, we use the Nose-Hoover (NH) thermostat in the framework of RESPA (REversible System Propagator Algorithm) (Tuckerman et al., 1992). The NH chain algorithm is based on a dynamical equation in an extended system (for simplicity, we omitted the index \( i \) that specifies the particle number),

\[
\frac{d\mathbf{r}}{dt} = \frac{\mathbf{p}}{m}
\]

\[
\frac{d\mathbf{p}}{dt} = \mathbf{f} - \frac{p_{\eta 1}}{Q} \mathbf{p}
\]
3.3. MOLECULAR MOVE ALGORITHM

\[ \frac{d\eta_1}{dt} = \frac{p_{\eta_1}}{Q_1} \]

\[ \frac{dp_{\eta_1}}{dt} = \left( \sum \frac{p^2}{m} - 3NkT \right) - \frac{p_{\eta_2}p_{\eta_1}}{Q_2} \equiv F_\eta \]

\[ \frac{d\eta_2}{dt} = \frac{p_{\eta_2}}{Q_2} \]

\[ \frac{dp_{\eta_2}}{dt} = \left( \frac{p_{\eta_2}^2}{Q_\eta_1} - kT \right) - \frac{p_{\eta_3}p_{\eta_2}}{Q_{\eta_3}} \]

\[ \ldots \]

\[ \frac{dp_{\eta_M}}{dt} = \left( \frac{p_{\eta_M}^2}{Q_{\eta_M-1}} - kT \right) \]

where \((r, p)\) are the coordinates and momenta of particles (of the physical system) and \((\eta_i, p_{\eta_i})\) (where \(i=1,2,\ldots, M\)) are sets of extended coordinates and momenta that are introduced to realize canonical ensembles of the physical system. \(Q_i\) is a mass for the corresponding variables.

When we introduce the RESPA algorithm for the above dynamical equation, we obtain a series of update procedure from time \(t\) to \(t + h\),

\[ p_{\eta_1} = p_{\eta_1}(t) + \frac{h}{2} F_\eta(t) \]

\[ \eta_1 = \eta_1(t) + \frac{h}{2} p_{\eta_1} \]

\[ v' = v(t) \exp \left( \frac{h}{2} p_{\eta_1} \right) \]

\[ r(t + h) = r(t) + h v \]

\[ v(t + h) = \left( v' + \frac{h}{2} f(t + h) \right) \exp \left( -\frac{h}{2} p_{\eta_1} \right) \]

\[ v(t + h) = v' \exp \left( -\frac{h}{2} p_{\eta_1} \right) \]

\[ \eta_1(t + h) = \eta_1 + \frac{h}{2} p_{\eta_1} \]

\[ p_{\eta_1}(t + h) = p_{\eta_1} + \frac{h}{2} F_\eta(t + h) \]

for a single-layer NH algorithm. For a NH chain, one adds the corresponding lines at the top and bottom of this procedure.

To realize the similar stability of the ensemble, \(Q_1\) should scale with the number of mass particles as well as the temperature, while other \(Q_i\)s should be independent of the size, but are much less important. Inside CafeMol, the single value of mass for the thermal particles, \(Q\), is used and its value is defined as “csmass_per” (the mass per number of particles) times the total number of mass-particles. The default value of csmass_per is set as 1.0 in the “general.para” file, and can be changed in “<<<< redefine para”.
3.3.4 Langevin MD

*CafeMol* equips the underdamped Langevin dynamics represented by

$$ m_i \frac{d^2 r_i}{dt^2} = f_i - m_i \gamma_i \frac{dr_i}{dt} + m_i \xi_i = f_{\text{total},i} (3.45) $$

where $m_i$ is the mass, $f_i$ is the force derived from the total potential energy function $V$ as $f_i = -\frac{\partial V}{\partial r_i}$, $\gamma_i$ is the friction coefficient, and $\xi_i$ is the random force (divided by the mass) The latter is a white and Gaussian random force with the mean $\langle \xi_i(t) \rangle = 0$ and the variance $\langle \xi_i(t)\xi_i(t') \rangle = 2m_i^{-1} \gamma_i k_B T \delta(t-t') \mathbf{1}$, where the bracket means the canonical ensemble average, and $\mathbf{1}$ is a 3×3 unit matrix.

For time integration, *CafeMol* uses a simple algorithm developed by Honeycutt-Thirumalai (Honeycutt and Thirumalai, 1992) (or Guo-Thirumalai (Guo and Thirumalai, 1995)). Based on the velocity version of the Verlet algorithm, this formula is derived by the Taylor expansion about $\gamma h$ and thus is valid for small $\gamma h$ case. The coordinate update in Verlet formula is

$$ r_i (t+h) = r_i(t) + v_i(t)h + \frac{1}{2} h^2 \frac{f_{\text{total},i}(t)}{m_i} (3.46) $$

Replacing $f_{\text{total},i}(t)$ with its three components, we get,

$$ r_i (t+h) = r_i(t) + v_i(t)h \left( 1 - \frac{\gamma_i h}{2} \right) + \frac{1}{2} h^2 \left( \frac{f_i(t)}{m_i} + \xi_i(t) \right) (3.47) $$

This is the formula for the coordinate updates in *CafeMol*. For velocity update, the original Verlet formula gives

$$ v_i(t+h) = v_i(t) + \frac{f_{\text{total},i}(t) + f_{\text{total},i}(t+h)}{2m_i} h (3.48) $$

where $f_{\text{total},i}(t+h)$ depends on the velocity at $t+h$, and thus the formula need to be solved self-consistently. We can only derive the closed formula for small $\gamma h$ limit. Namely, we can get

$$ v_i(t+h) \left( 1 + \frac{\gamma_i h}{2} \right) = v_i(t) \left( 1 - \frac{\gamma_i h}{2} \right) + \frac{h}{2} \left( \frac{f_i(t)}{m_i} - \frac{f_i(t+h)}{m_i} + \xi_i(t) + \xi_i(t+h) \right) (3.49) $$

Using the approximation

$$ \left( 1 + \frac{\gamma_i h}{2} \right)^{-1} = 1 - \frac{\gamma_i h}{2} + \left( \frac{\gamma_i h}{2} \right)^2 + ... (3.50) $$

we end up with the formula,

$$ v_i(t+h) = v_i(t) \left( 1 - \frac{\gamma_i h}{2} \right) \left[ 1 - \frac{\gamma_i h}{2} + \left( \frac{\gamma_i h}{2} \right)^2 \right] + \frac{h}{2} \left( 1 - \frac{\gamma_i h}{2} \right) \left( \frac{f_i(t)}{m_i} - \frac{f_i(t+h)}{m_i} + \xi_i(t) + \xi_i(t+h) \right) (3.51) $$

where we note that the first line contains a third order term, which can be omitted, but is harmless too. *CafeMol* exactly uses the above formula for the velocity update. The random force $\xi_i(t)$ in finite time step is drawn from the Gaussian (normal) distribution with its mean

$$ \langle \xi_i(t) \rangle = 0 (3.52) $$

and its variance

$$ \langle \xi_i(t)\xi_i(t') \rangle = \frac{2\gamma_i k_B T}{m_i} \delta_{t,t'} \mathbf{1} (3.53) $$

Although, the Langevin dynamics has no problems that we raised in the section on constant-temperature Newtonian dynamics, there are some drawbacks in Langevin dynamics, too. First, while simulating protein
3.3. MOLECULAR MOVE ALGORITHM

folding, we observed that folding is less cooperative near the transition state and thus is less efficient. For some proteins, we could simulate reversible folding only with Newtonian dynamics, but not with Langevin dynamics. Another limitation of Langevin dynamics in its current form is its ignorance of hydrodynamics interaction. It has been known that ignorance of hydrodynamic interaction slows down translational and rotational motion of macromolecules represented as collection of many small particles. Yet, for general use in simulations of biomolecular complex, Langevin dynamics is the first choice in CafeMol, at the moment.

In CafeMol, $h$ is called “tstep.size”, and its value has to be given in “<<< md_information”. Typical values in Newtonian dynamics would be 0.05 ~ 0.2. Normally, we can use slightly larger value for the Langevin dynamics than for Newtonian dynamics. Empirically, when nucleic acids are involved, the time step should be somewhat smaller.

Choice of the friction coefficient $\gamma$ value is a matter of concern. First of all, if we literaly use the viscosity of water at physiological condition and set the radius of an amino acid as 3.8\text{Å}, the Stokes law gives an estimate of the friction coefficient being 4.5 in the CafeMol unit. Yet, this is not necessarily the best choice. As mentioned above, the current simple form of Langevin dynamics ignores the hydrodynamic effect, which tends to slow down collective motions. The larger scale motion would be slowed down more significantly. Thus, if we use the friction coefficient value of 4.5 determined by the estimate of one amino acid, collective dynamics including folding, conformational change, and overall translational and rotational dynamics would be highly slowed down. Just empirically, we found a smaller value such as $\gamma = 0.25$ works reasonable for folding of single-domain protein. Even larger scale dynamics such as diffusion of a large protein can be simulated with even smaller value such as $\gamma = 0.02$. Note, however, that a smaller value could induce instability of the trajectory. Depending of the choice of the value, the absolute time scale changes drastically, but we hope qualitative arguments such as pathways are robust and thus would not be so different by the different value of $\gamma$. Inside CafeMol, the friction coefficient $\gamma$ is called “fric const”. Its default value is set in general, para as 0.25, and can be changed in the input-block “<<< redefine para”. To simulate larger scale dynamics of large molecular systems, one may want to reduce it further (it has been tested as small as 0.02).

3.3.5 Simple minimization
to be written

3.3.6 MPC (Multi-Particle Collision) dynamics

CafeMol equips the MPC (Multi-Particle Collision) dynamics to incorporate the hydrodynamic effect (Gompper et al., 2009). In this dynamics, the solvent is modeled by a large number $N_s \equiv \gamma_{n_x n_y n_z}$ of point-like particles of mass $m_s$ which move in continuous space with a continuous distribution of velocities. This algorithm proceeds in two steps.

In the first of steps, a free-streaming step, the positions of the solvent particles at time $t$, $r_i^{(s)}(t)$, are updated simultaneously according to

$$r_i^{(s)}(t + \tau_{col}) = r_i^{(s)}(t) + v_i^{(s)}(t) \tau_{col}, \quad (3.54)$$

where $v_i^{(s)}(t)$ is the velocity of the solvent particle and $\tau_{col}$ is the value of the time interval between collision steps $(\tau_{col} = ci \times h)$ where $ci$ is the step number of collision interval and $h$ is the time step for integration. (the detail will be mentioned later.) During this first steps ($\tau_{col}$), the time evolution of the solute system particles such as those in proteins, DNAs, and RNAs is calculated by the Newtonian dynamics. For time-integration, CafeMol uses the velocity-version of Verlet algorithm.

The second part of the algorithm is the collision step which is executed on both solvent particles and system (solute) beads. This procedure is performed for every $ci$ MD step (every $\tau_{col} = ci \times h$). The system is divided into a cubic grid with $(n_x \times n_y \times n_z)$ cells, which have $a_x(= L_x/n_x), a_y(= L_y/n_y), a_z(= L_z/n_z)$ sides where $(L_x, L_y, L_z)$ is the sides of whole simulation box and $(n_x, n_y, n_z)$ is the number of grid for each side. There is no restriction on the total number of solvent particles in each grid although the total number of solvent particles ($N_s = \gamma_{n_x n_y n_z}$) in the system is conserved, where $\gamma_s$ is the average number of solvent particles per cell. Stochastic multiparticle collisions are performed within each individual cell by rotating the velocity of each particle relative to the center-of mass velocity $v_{c,m.}(t)$ of all the particles within that cell,
\begin{equation}
\mathbf{v}_i(t + \tau_{\text{col}}) = \mathbf{v}_{c.m.}(t) + \mathbf{R}(\mathbf{v}_i(t) - \mathbf{v}_{c.m.}(t)),
\end{equation}

where \( \mathbf{R} \) is a rotation matrix which rotates velocities by a fixed angle \( \alpha \) \((0 < \alpha \leq \pi)\) around an axis generated randomly for each cell and at each step. The aim of the collision step is to transfer momentum between the particles while conserving the total momentum and energy in each cell.

Ihle and Kroll (Ihle and Kroll, 2001) pointed out that at low temperatures the transport coefficients of the MPC dynamics solvent show anomalies. Ihle and Kroll showed that it is possible to greatly improve the behavior of the algorithm by placing the cubic grid in a random position at each collision step. In practice, the easiest way to implement such a grid shift is to move all the particles with the same random vector with components in the interval \([-a/2, a/2]\) before each collision step. After the collision step the particles are returned to their original positions. (This treatment is implemented in CafeMol).

To give (sustain) a temperature \( T \) the solvent velocities are rescaled after each collision step (Kikuchi et al., 2003).

### 3.4 Simulation protocol

#### 3.4.1 Constant temperature MD

The constant temperature MD is the most standard simulation protocol in CafeMol. Currently, one can use either constant-temperature “Newtonian” dynamics or Langevin dynamics.

#### 3.4.2 Searching \( T_F \)

CafeMol uses a bisection method to automatically compute the folding transition temperature \( T_F \). Namely, you first specify, in the input file, the lower and upper bounds of \( T_F \). CafeMol first simulates the protein at the mid-point temperature of the upper and lower bounds and see if the protein is near native conformation for more than half of the simulated time. If yes, this temperature is set as the new lower bound of \( T_F \). Otherwise, the simulated temperature was set as the new upper bound of \( T_F \). With the new set of the lower and the upper bounds, CafeMol repeats the simulation. This iteration lasts for the required times to narrow the range. For an acceptable estimate of \( T_F \), the MD step number needs to be larger than the folding and unfolding time scale near \( T_F \). Typically, proteins with less than 100 amino acids are easy, but those with more than 150 residues are almost impossible to achieve this condition. Thus, CafeMol allows the auto \( T_F \) search only for a single chain protein with less than 150 residues.

With the default parameter set, \( T_F \) for several small proteins were computed, and these \( T_F \) range from 330K to 390K.

#### 3.4.3 Simulated annealing

Simulated annealing is to search the lower (possibly lowest) energy structure in the simulated system. It performs MD with temperature gradually decreasing. Basically, it uses the constant-temperature MD routine and the temperature there is decreased at a certain frequency. See the next chapter for details.

#### 3.4.4 Switching Go potential

Many biomolecules, most prominently proteins, work by changing their conformations depending on their interactions with their partner molecules. For example, proteins change their conformation upon binding to ligands. How binding and conformational change are coupled in proteins is in itself a subject to be studied. However, for studying more complex biological phenomena, we may want to enforce such a conformational change by hand. Switching Go model is proposed for this type of simulations.

In switching Go model, we first simulate a protein by the Go model \( V_{\text{Go}}(\mathbf{R} | \mathbf{R}_A) \) with a structure \( A \) being the reference structure. The protein usually resides nearby the structure \( A, \mathbf{R} \sim \mathbf{R}_A \).
we suddenly change the reference structure of the Go model to another structure B, resulting in a new Go potential $V_{Go}(\mathbf{R}|\mathbf{R}_B)$. The protein jumps from the bottom of the Go model $V_{Go}(\mathbf{R}|\mathbf{R}_A)$ to the uphill slope of the new Go potential $V_{Go}(\mathbf{R}|\mathbf{R}_B)$. Note that the simulated structure $\mathbf{R}$ does not change suddenly, but changes continuously. Right after the switch, the protein relaxes its conformation from $\mathbf{R} \sim \mathbf{R}_A$ to $\mathbf{R} \sim \mathbf{R}_B$. This mimics the conformational change from the structure A to the structure B, in a simple way.

_CafeMol_ is ready to switch Go potential in any fixed times.

### 3.4.5 Switching bias in multiple basin potential

Switching Go model is, perhaps the simplest way to realize some large-scale conformational change in biomolecules. However, in switching Go model, the protein is “excited” to the new potential and the resulting procedure is nothing but the relaxation on the new Go potential surface. This resembles to the photo-activated process. Biologically, however, many events are thermally activated and thus overcoming energy barrier by thermal fluctuation may be of essential importance, in some cases. To realize this thermally activated conformational transition, _CafeMol_ uses the multiple-basin model. We first simulate a protein with the multiple-basin model with the structure A and B being two basins. $V_{MB}(\mathbf{R}|\mathbf{R}_A,\mathbf{R}_B,\Delta V)$ where $\Delta V$ is positive and sufficiently large so that the basin A is more stable. The protein mostly resides in the basin $\mathbf{R} \sim \mathbf{R}_A$. At a certain MD step, we suddenly change the bias $\Delta V$ to a negative and sufficiently large absolute value so that the basin B is now more stable. Soon after the switch, the protein would still fluctuate in the basin A, but after a while, it overcomes the barrier to reach the more stable basin B, through thermal fluctuation.

We can couple the switch bias in MBP and switch in the reference structure as in the case of switching Go model.

### 3.4.6 Replica exchange method

The replica exchange method, also known as the parallel tempering method, has been frequently used to improve sampling efficiency of complex systems (Hukushima and Nemoto, 1996). One prepares many copies of the system of interest where each replica evolves with different temperatures. For studies on biological molecules, the method was applied to molecular dynamics simulation which is often called as replica exchange molecular dynamics (REMD) (Sugita and Okamoto, 1999).

Originally the method was proposed as the replica variable being temperature, but soon it was generalized to other parameters in the Hamiltonian. This advanced method is called as multidimensional REM or Hamiltonian REM (Sugita et al., 2000; Fukunishi et al., 2002). As indicated by the term “multidimensional”, it is possible to use two or more replica variables simultaneously.

The method would be useful to determine the folding temperature of larger proteins, calculate the potential of mean force along a reaction coordinate, and investigate any other thermodynamics quantities.

Here, we introduce briefly theoretical basis and procedure of REMD. (See ref. (Sugita et al., 2000) for details.) For generality, a case of two-dimensional REMD (one dimension is temperature space, the other is certain parameter space) is considered.

At the beginning of REMD simulation, a target system is replicated to some or many “replica”s which have exactly the same property except the replica variables (e.g. temperature, ionic strength, etc.).

The Hamiltonian of the $i$-th replica is

$$H_{m(i)}(\mathbf{R}^{[i]},\mathbf{p}^{[i]}) = K(\mathbf{p}^{[i]}) + V_{\lambda_m(i)}(\mathbf{R}^{[i]}). \quad (3.56)$$

at the $m$-th inverse temperature

$$\beta_{m(i)} = \frac{1}{k_B T_{m(i)}}. \quad (3.57)$$

where $K$ is the kinetic energy, $V$ is the potential energy, $\lambda_m$ is the $m$-th parameter for potential energy, $k_B$ is Boltzmann constant, and $T_m$ is $m$-th temperature. Here, the index $m$ indicates a set of replica variables.

$$\Lambda_m = (T_m, \lambda_m) (m = 1, 2, ..., N_{rep})$$
Because replicas never interact physically each other, the joint distribution probability of the entire system is represented as the product of each canonical system.

\[
P_{REM} = \prod_i N_{rep} P_i(R_i)
\]

\[
P_{REM} = \frac{1}{Z_{REM}} \exp\left(-\sum_i \beta_{m(i)} H_{m(i)}(R_i, p[i])\right).
\]

(3.58)

Then those replicas are simulated simultaneously and independently, and by a certain frequency, they try to exchange replica variables each other. Now we suppose replica \(i\) and \(j\) exchange their parameter set \(m\) and \(n\), respectively. The transition probability is introduced as

\[
W(R_i^i, \Lambda_m; R_j^j, \Lambda_n).
\]

Considering the detailed balance condition, we have

\[
P_{REM}(..., R_i^i, \Lambda_m; ..., R_j^j, \Lambda_n; ...) W(R_i^i, \Lambda_m; R_j^j, \Lambda_n) = P_{REM}(..., R_j^j, \Lambda_n; ..., R_i^i, \Lambda_m; ...) W(R_j^j, \Lambda_n; R_i^i, \Lambda_m)
\]

(3.59)

which directly leads to

\[
W(R_i^i, \Lambda_m; R_j^j, \Lambda_n) \over W(R_i^i, \Lambda_m; R_i^i, \Lambda_m) = \exp(-\Delta)
\]

(3.60)

where we define

\[
\Delta \equiv \beta_m (V_{\lambda_m}(R_j^j) - V_{\lambda_m}(R_i^i)) - \beta_n (V_{\lambda_n}(R_j^j) - V_{\lambda_n}(R_i^i))
\]

(3.61)

One reasonable choice which satisfies the above equation is the Metropolis-type criteria,

\[
\begin{cases}
W(R_i^i, \Lambda_m; R_j^j, \Lambda_n) = 1 & \text{for } \Delta \leq 0 \\
W(R_i^i, \Lambda_m; R_j^j, \Lambda_n) = \exp(-\Delta) & \text{for } \Delta \geq 0.
\end{cases}
\]

Whenever the exchange is accepted, momenta should be rescaled as,

\[
\begin{align*}
p_i^i &\equiv \sqrt{T_m/T_n} p_i^i \\
p_j^j &\equiv \sqrt{T_m/T_n} p_j^j.
\end{align*}
\]

(3.62)

### 3.4.7 Feedback-optimized REMD

*CafeMol* implements the feedback-optimized REMD that can iteratively optimize the temperature distribution to achieve one of the best performance in sampling (Trebst et al., 2006; Katzgraber et al., 2006)

### 3.4.8 Modified multicanonical ensemble method

*CafeMol* implements the modified multicanonical ensemble method (Gosavi et al., 2006) proposed for efficient sampling in structure-based protein simulations. The modified multicanonical ensemble method is a kind of umbrella sampling methods with an optimally tuned Gaussian potential added to the physical potential. The potential well to be added onto the original Hamiltonian is

\[
V(E) = -E_{\text{depth}} \exp[-(E - E_{\text{mid}})^2/(2\sigma^2)],
\]

(3.63)

where \(E\) is the (physical) potential energy, and \(E_{\text{depth}}, E_{\text{mid}}, \) and \(\sigma\) are parameters to be tuned. In the study of protein folding problem, for each of target proteins, one needs to calibrate these parameters carefully to compensate the free energy barrier of the original system around the transition state. In doing so, the energy barrier of folding is substantially reduced and the reversible folding-unfolding process can be simulated. The thermodynamics of the original Hamiltonian system can be regained using a reweighting factor, \(\exp[V(E)/k_B T]\), where \(k_B\) is the Boltzmann’s constant and \(T\) is the simulation temperature.

See “<<< modified_muca” in the subsequent chapter.
Chapter 4

Files and data format

The input/output of CafeMol is realized through several files that have different formats. Specifically, for input, there are four kinds of files, namely “PDB file”, “native-info file”, “input file”, and “parameter file”, respectively. (For the AICG model, one needs some more inputs. See the AICG-related sections) On the other hand, for output, the following formats are used: “PDB file”, “movie file”, “CARD coordinate”, “CARD velocity”, “DCD coordinate”, “DCD velocity”, “data file”, “time-series file”, “native-info file”, ”protein structure file”, and ”replica information file”. (For GNU compiler, “DCD” format is not supported.) Note that for a particular simulation not all the files are needed or generated. Moreover, for the purpose, you have multiple options of file format to choose.

CafeMol implements several tools to convert one file format to the other. For example, using crd2pdb and pdb2crd one can convert the format between “PDB” and “CARD coordinate”. Similarly, movie2dcd and dcd2movie are used to convert the format between “movie” and “DCD coordinate”. Except for PDB, CARD, and DCD format, the same general rules as described in Chapter 5 are applied.

In this chapter, we explain one by one the data format of above files, as well as their basic purpose. We describe the format in two ways. One simply indicates the data-type by using some abbreviations, e.g. “int” for integer, “real” for real, and “char” for character; it is used for free-column-width or binary data format. The other employs the FORTRAN-style description of formatted output, for fixed-column-width data.

4.1 Input files

4.1.1 PDB file (.pdb)

The standard PDB (Protein Data Bank) format is adopted for providing the information of “sequence”, “initial conformation”, and “native structure”. Typically you may have multiple PDB files and you can specify the purpose of each file by configuring the “input file” (See Chapter 5). Please access the website http://www.wwpdb.org/docs.html for an accurate explanation of PDB format.

4.1.2 Native-info file(.ninfo)

Although the simplest way to define the native structure information is to use the PDB file (see above), the native-info file provides an alternative way to describe the native structure, which is more flexible. For switching-Go simulation, in particular, this type of input is mandatory. Typically, this file is originally generated automatically by a previous simulation of the related system, and you may need to edit these files for your final target system. It is also possible, in principle to prepare it by yourself.

Note that CafeMol allows two styles of native-info input: all-in-one-file style and one-by-one-file style. For the former, CafeMol reads native-structure information of all the system from one file. In this case, the absolute amino acids indices, i.e. imp1, imp2, etc., are used for indicating the mass-point numbers as input data. For the case of one-by-one-file style, you need to prepare native-info files separately for every intra- and inter-units. In this case, local interaction sets (bond length, bond angle, and dihedral angle) are required only in the intra-unit files and it reads intra-unit indices such as imp1un, imp2un, etc.
The data in a native-info file are managed into blocks, as shown below (data-types are indicated in parenthesis):

1. Bond length

```
<<<<< native bond length
** coef_bd(kcal/mol) = factor_bd * correct_bd_mgo * cbd * energy_unit_protein
** ibd  iunit1  iunit2  imp1  -imp2  imp1un-imp2un
bond ibd iunit1 iunit2 imp1 imp2 imp1un imp2un
(a4, 7(1x6),
......
>>>>>
```

Each row starting with a key word “bond” describes one bond, in which `iunit1` and `iunit2` are the unit numbers that this bond belongs to (these two are always the same!), `ibd` is the bond number, `imp1` and `imp2` are the indices of N- and C-terminal amino acids, respectively, `imp1un` and `imp2un` are the intra-unit indices of the same two amino acids, and `bd_nat` is the native length of the bond. Typically, there are also comment lines that start with asterisks (“*”).

In the energy function of bond length, the coefficient is expressed as a product of a number of parameter

\[
\text{coef}_{bd}(\text{kcal/mol}) = \text{factor}_{bd} \times \text{correct}_{bd\_mgo} \times \text{cbd} \times \text{energy\_unit\_protein}
\]

where `factor_{bd} = local\_unit \times factor_{sec\_bd}`, `local\_unit` is overall weight of the local interaction for a unit, `factor_{sec\_bd}` is the weight related to secondary structure (not implemented in current version), and `correct_{bd\_mgo}` is for the weakening of local strain energy on some particular sites when doing multiple-basin Go simulation. Both `factor_{bd}` and `correct_{bd\_mgo}` are site-specific parameters. `cbd` and `energy\_unit\_protein` are constant coefficients of the energy function, for which you can find the explanation in Chapter 3. At last, `bdtype` indicates the bond type: ‘pp’ means bond within protein, blank means bond within DNA and ligand, and other strings are used for various RNA bonds (See 3.2.6).

As mentioned above, usage of columns is somewhat different between all-in-one-file style and one-by-one-file style. For the all-in-one-file case, the 5th and the 6th columns are read as input data, while for the one-by-one-file case, the 7th and the 8th columns are read as input data. This means that, in either case, there are some columns that are not used. Still you should keep these unused columns as they are. Also noted is that for the one-by-one-file case, this block should be in the intra-unit native-info files.

2. Bond angle

The data block for bond angle term is stored as following:

```
4.1. INPUT FILES

native bond angles
** coef_{ba}(\text{kcal/mol}) = \text{factor}_{ba} \times \text{correct}_{ba \_mgo} \times \text{cba} \times \text{energy}_{unit \_protein}

where

coef_{ba}(\text{kcal/mol}) = \text{factor}_{ba} \times \text{correct}_{ba \_mgo} \times \text{cba} \times \text{energy}_{unit \_protein}

In this case, factor_{ba} = \text{local \_unit} \times \text{factor \_sec \_ba}, in which factor \_sec \_ba is not implemented yet. The meaning of parameters is similar to those of bond length. Note that there are three relevant amino acids to define a bond angle. The \text{iunit1} and \text{iunit2} are the unit numbers that the bond angle belongs to (these two are always the same!). In the last column, the batype is a string to indicate the bond angle type: ‘ppp’ means bond angle within protein, blank means bond angle within DNA and ligand, and all other strings are used for RNA bond angles (See 3.2.6).

The last paragraph in “Bond length” subsection applies here as well.

3. Dihedral angle

The data for dihedral angle term are managed to be:

native dihedral angles
** coef_{dih1} = factor_{dih} \times \text{correct}_{dih \_mgo} \times \text{cdih}_{1} \times \text{energy}_{unit \_protein}

** coef_{dih3} = factor_{dih} \times \text{correct}_{dih \_mgo} \times \text{cdih}_{3} \times \text{energy}_{unit \_protein}

** idih iunit1- iunit2 imp1- imp2- imp3 imp4 imp1un-dihd idih iunit1 iunit2 imp1 imp2 imp3 imp4 imp1un
(a4, 9(1x6)),

...
CHAPTER 4. FILES AND DATA FORMAT

where

\[
coef_{dih,1}(\text{kcal/mol}) = factor_{dih} \times correct_{dih, homo} \\
\times cdih_{1} \times energy_{unit, protein}
\]

and

\[
coef_{dih,3}(\text{kcal/mol}) = factor_{dih} \times correct_{dih, homo} \\
\times cdih_{3} \times energy_{unit, protein}
\]

are coefficients for the $2\pi$-period and $\frac{2}{3}\pi$-period terms of the energy function, respectively. Similarly, you can find the meaning of each parameter by looking up above explanation for the bond length. Note that four amino acids are needed to define a dihedral angle. In the last column, dihtype indicates the dihedral angle type: 'pppp' means dihedral angle within protein, blank means dihedral angle within DNA and ligand, and other strings are used for various RNA dihedral angles (See 3.2.6).

The last paragraph in “Bond length” subsection applies here as well.

4. **Native contact**

The information of native contact is described as following:

```plaintext
<<<<< native contact
** total_contacts = ncon (i6)
** definition_of_contact = dfcontact (10:2) A
** coef_go(kcal/mol) = factor_go * icon * dummy_go * cgol1210 * energy_unit_protein
** contact between unit iunit1 (i6) and iunit2 (i6)
** total_contact_unit = ncon_unit (i6)
** icon iunit1- iunit2 imp1- imp2 imp1un- imp2un
contact icon iunit1 iunit2 imp1 imp2 imp1un imp2un
(a7, 7(1xii6),
....
>>>>>
```

```plaintext
coef_go, factor_go, dummy_go
coef_go, contype
2(1xf12:4),
(1x6), (1xf12:4), a4)
```
For most parameters, you can find counterparts in the case of local interaction mentioned above. Here are some specialties:

\[
\text{coef}_\text{go}(\text{kcal/mol}) = \text{factor}_\text{go} \times \text{dummy}_\text{mgo} \\
\times \text{go1210} \times \text{energy}_\text{unit}_\text{protein}
\]

where \( \text{factor}_\text{go} = \text{go}_\text{unit} \times \text{factor}_x \), \( \text{go}_\text{unit} \) is overall weight of the non-local interaction within a particular unit (or between a pair of units for the case of inter-unit contact), and \( \text{factor}_x \) is a site-dependent weight which is not implemented in current version of CafeMol.\( \text{dummy}_\text{mgo} \) is a boolean parameter indicating real contact (=1) or dummy-contact (=0). Please see Chapter 3 for detailed explanation.\( \text{go1210} \) and \( \text{energy}_\text{unit}_\text{protein} \) are constant coefficients of the energy function as mentioned above. In the last column, \( \text{contype} \) indicates the contact type: ‘p-p’ means contact between proteins, blank means contact for DNA-DNA, protein-DNA, and protein-ligand interactions, and other strings are used for various contact types related to RNA (See 3.2.6).

5. Native basepair

The information for the native basepair in RNA model is described as following:

```
<<<< native basepair
** total_contact = ncon (i6)
** definition_of_contact = dfcontact (i10:2) A
** coef_go(kcal/mol) = factor_go * icon * dummy_mgo * cbp1210 * energy_unit_protein
** contact between unit iunit1 (i6) and iunit2 (i6)
** icon iunit1- iunit2 imp1- imp2 imp1un- imp2un
basepair icon iunit1 iunit2 imp1 imp2 imp1un imp2un
(a8, 7(1xi6), ....

```
## CHAPTER 4. FILES AND DATA FORMAT

For most parameters, you can find counterparts in the case of native basepair mentioned above. The difference is $cst1210$, which is a constant particular for the base-stacking interaction in RNA model, and $bstype$ the type of base stacking in RNA model (See 3.2.6).

We note that if $coef_{go} < 1.0 \times 10^{-6}$, it is regarded as zero and this contact is removed and the standard excluded volume interaction is turned on.

Again, we emphasize that CafeMol reads the absolute amino acids indices, i.e. $imp1$ and $imp2$ for the case of all-in-one-file style, while it reads intra-unit indices $imp1un$, and $imp2un$ for the case of one-by-one-file style.

### 4.1.3 Input file (\*.inp)

The input file is the central file for users to configure a simulation. Please see Chapter 5 for a detailed explanation.

### 4.1.4 Parameter file (\*.para)

CafeMol reads the default parameters of the energy function from several parameter files listed below. These files are placed under the directory “./para”, by default, and we recommend that they remain unchanged. User can change most of these value in the input file.

The content of each file is roughly indicated by its name. All the files are made obeying the general rules (with exceptions, though):
In the following we briefly describe the meaning of parameters contained in each file.

1. general.para

```
<<< para_cafemol_gen
  name_of_parameter = value
  ....
>>>>
```

where the “name_of_parameter” can be one of the following variables.

- velo_adjst: the coupling parameter of Berendsen thermostat (1.0 by default).
- csmass_per: the mass of thermal particle per bead of Nose-Hoover thermostat (1.0 by default).
- csmass_mpc_per: the mass of thermal particle per MPC solvent of Nose-Hoover thermostat (1.0 by default).
- rneighbor_dist: the cutoff distance to define a neighbor (24.0 by default).
- rmass: default value of particle mass (10.0 by default).
- fric_const: friction constant for Langevin dynamics simulation (0.25 by default).

2. protein.para

```
<<< para_cafemol_pro
  name_of_parameter = value
  ....
>>>>
```

The variables include

- energy_unit_protein: the energy unit for protein modeling (kcal/mol, unity by default).
- cbd: constant coefficient of the energy function for bond length. (100 by default)
- cba: constant coefficient of the energy function for bond angle. (20 by default)
- cdih_1: constant coefficient of the 2\pi-period term of the energy function for dihedral angle (1 by default)
- cdih_3: constant coefficient of the 2\frac{3\pi}{2}-period term of the energy function for dihedral angle (0.5 by default)
- n_sep_dlocal: the minimum number of amino acids that separate a non-local pair. (4 by default)
- n_sep_contact: the minimum number of amino acids that separate a contact pair. (4 by default)
- cutoff_go: the (relative) truncation distance for computing non-local Go interaction: When \( r_{ij} > \text{cutoff}_\text{go} \times r_{ij,0} \), then the Go energy is set to zero. (2.5 by default)
- cutoff_exvol: the (relative) truncation distance for computing non-local non-native repulsion: When \( r_{ij} > \text{cutoff}_\text{exvol} \times \text{cdist}_\text{rep12} \), then the repulsive energy is set to zero. (2.0 by default)
- dfcontact: the cutoff distance to define the native contact (6.5 by default).
- cgo1210: constant coefficient \( \varepsilon_{go} \) of the energy function for non-local Go interaction. (0.3 by default)
- cdist_rep12: reference distance \( d \) in the non-native repulsive interaction of Clementi et al.’s Go model. (4 by default)
- crep12: constant coefficient \( \varepsilon_{ev} \) in the non-native repulsive interaction of Clementi et al.’s Go model. (0.2 by default)
3. dna.para
   To be added

4. rna.para
   To be added

5. lipid.para
   To be added

6. ligand.para
   In ligand.para file, the parameters for the local and non-local interactions of explicit ligand are defined (See Chapter 3). It contains a input block of the following standard form

   <<<< para cafemol_lig
   name_of_parameter = value
   ..... 
   >>>>>

   The variables include
   
   *energy unit lig*: the energy unit for explicit ligand and hydrophobic interaction modeling (kcal/mol, unity by default).
   
   *cbd_lig*: constant coefficient of the energy function for bond length. (120 by default)
   
   *cba_lig*: constant coefficient of the energy function for bond angle. (24 by default)
   
   *cdih_lig*: constant coefficient of the harmonic improper dihedral angle (24 by default)
   
   *cutoff_{exvol}lig*: the (relative) truncation distance for computing non-local non-native repulsion: When $r_{ij} > cutoff_{exvol}lig \times d_l$, then the repulsive energy is set to zero. (2.0 by default)
   
   *cdist_{rep12_lpro}*: reference distance $d_l$ in the non-native repulsive interaction between ligand and protein (4.6 by default)
   
   *cdist_{rep12_llig}*: reference distance $d_l$ in the non-native repulsive interaction between ligand and ligand (4.6 by default)
   
   *crep_{12_lig}*: constant coefficient $\varepsilon_{ev,l}$ in the non-native repulsive interaction. (0.2 by default)

7. aicg_genetic.para
   The aicg_genetic.para provides the generic parameters for both the AICG1 and AICG2 simulations. When AICG1 was used (See Chapter 5) with the option $i_{aicg} = 1$, the default AICG1 parameters derived by fluctuation matching are specified in the block “<<<< aicg_para”. We briefly describe the meaning of these parameters.

   <<<< aicg_para
   name_of_parameter = value
   ..... 
   >>>>>

   The name_of_parameter includes
   
   *cbd_{aicg}*: constant coefficient of the energy function for bond length
   
   *cba_{aicg}_{G}*: constant coefficient of the energy function for bond angle containing glycine
   
   *cba_{aicg}_{H}*: 
constant coefficient of the energy function for bond angle in helix  
\( c_{\text{cba}_{\text{aicg}}E} \):

constant coefficient of the energy function for bond angle in \( \beta \)-strand  
\( c_{\text{cba}_{\text{aicg}}T} \):

constant coefficient of the energy function for bond angle in turn  
\( c_{\text{cba}_{\text{aicg}}C} \):

constant coefficient of the energy function for bond angle in random coil  
\( c_{\text{dih}_{\text{aicg}}H} \):

constant coefficient of the energy function for dihedral angle in helix  
\( c_{\text{dih}_{\text{aicg}}E} \):

constant coefficient of the energy function for dihedral angle in \( \beta \)-strand  
\( c_{\text{dih}_{\text{aicg}}T} \):

constant coefficient of the energy function for dihedral angle in turn  
\( c_{\text{dih}_{\text{aicg}}C} \):

constant coefficient of the energy function for dihedral angle in random coil  
\( i_{\text{flag}_{\text{scale}}} \):

option for scaling the all-atom contact energies to heterogeneous coefficient of the energy function for nonlocal interactions.  
\( \text{ave}_{\text{caicg}} \):

average strength of the energy function for nonlocal interactions.  
\( \text{gen}_{\text{caicg}} \):

factor to scale the all atom contact energies into strength of the nonlocal interactions.  
\( \text{ecut}_{\text{low}_{\text{aicg}}} \):

lower cutoff of the contact energies.  
\( \text{ecut}_{\text{up}_{\text{aicg}}} \):

upper cutoff of the contact energies.

When AICG2 was used (See Chapter 5) with the option \( i_{\text{aicg}} = 1 \), the default AICG2 parameters derived by fluctuation matching are specified in the block “<<<< aicg2_para”.

```
<<<< aicg2_para
name_of_parameter = value
>>>>>
```

The name_of_parameter includes  
\( \text{cbd}_{\text{aicg}}2 \):

constant coefficient of the energy function for bond length  
\( \text{wid}_{\text{aicg}13} \):

Gaussian width of the local contact interactions for the \( i, i+2 \) residue pairs.  
\( \text{wid}_{\text{aicg}14} \):

Gaussian width of the local contact interactions for the \( i, i+3 \) residue pairs.  
\( i_{\text{flag}_{\text{scale}}_{\text{aicg}2}} \):

option for scaling the all-atom contact energies to heterogeneous coefficient of the energy function for local and nonlocal contacting interactions.  
\( \text{ave}_{\text{caicg}2_{\text{loc}}} \):

average strength of the energy function for the local contact interactions.  
\( \text{ave}_{\text{caicg}2_{\text{nloc}}} \):

average strength of the energy function for the nonlocal contact interactions.  
\( \text{gen}_{\text{caicg}2_{\text{loc}}} \):

factor to scale the all atom contact energies into strength of the local contact interactions.  
\( \text{gen}_{\text{caicg}2_{\text{nloc}}} \):

factor to scale the all atom contact energies into strength of the nonlocal contact interactions.  
\( \text{ecut}_{\text{low}_{\text{aicg}2}} \):

lower cutoff of the contact energies  
\( \text{ecut}_{\text{up}_{\text{aicg}}} \):

upper cutoff of the contact energies
8. electrostatic.para

The electrostatic.para defines parameters for the electrostatic interactions. First, the four parameters that specify the electrostatics are

\[
\begin{array}{ll}
\text{para cafemol ele} \\
cutoff & = \text{real (in } Å) \\
\text{ionic strength} & = \text{real (in molar/liter, 1M NaCl has its value 1)} \\
\text{diele water} & = \text{real (dimension-less ratio, for water at 300K this is about 78)} \\
i \text{diele} & = 0 \text{ (for constant dielectric constant) (no other value)} \\
\end{array}
\]

Second, in the same block, the following format is used to define charges for each type:

\[
\begin{array}{ll}
\text{para cafemol ele} \\
\text{CHARGE TYPE} AA ch arg e \\
\text{(char char real)} \\
\end{array}
\]

Each line defines the charge for the corresponding type of particles.

9. hydrophobic.para

The hydrophobic.para defines parameters for the hydrophobic interactions. This uses a hybrid style. First, it contains the lines such as

\[
\begin{array}{ll}
\text{para cafemol hp} \\
\text{name of parameter} = \text{value} \\
\end{array}
\]

The variables include

- \text{coef }_{HP}: the constant \(c_{HP}\) in the energy function of the hydrophobic interaction. \((0.58 \text{ by default})\)
- \text{coef }_{\rho \text{linear}}: the constant \(c_{\text{linear}}\) in the hydrophobic term. \((0.2 \text{ by default})\)
- \text{rho }_{\text{min}}: the constant \(\rho_{\text{min}}\) in the hydrophobic term. \((0.3 \text{ by default})\)

Second, in the same block, the following format is used to define hydrophobicity of 21 particle types (20 amino acids plus one non-amino acid):

\[
\begin{array}{ll}
\text{para cafemol hp} \\
\text{HPE TYPE} AA \text{ epsilon }_{HP} n_j n_{j,max} !j \\
\text{(char char real real real char int)} \\
\end{array}
\]

Each line defines the parameters for one amino acid or other hydrophobic particles. The first column is the keyword. The second column lists the amino acid name. The third, fourth, and fifth columns define the values for the constants \(\epsilon_{HP,A(i)}, n_A(j),\) and \(n_{\text{max},A(i)}\), respectively, in the energy function of the hydrophobic interaction. The last column is a comment, defining the index of each amino acid. The last column will not be read by the program, but is shown just for the convenience of remembering amino acid indices.
Third, in the hydrophobic.para file, there are two additional blocks. One that defines \( r_{\text{min},A(i),A(j)} \) and looks like

\[
<<<<\text{cutoff}\_dmin\_hp \\
\text{HPE\_CUTOFF} \quad \text{AA} \quad \text{value} \quad ... \\
(\text{char} \quad \text{char} \quad \text{real} \quad ...) \\
... \\
>>>>
\]

In the block, every three lines (9 columns in each line) define the length parameters for one amino acid or other hydrophobic particles. Within each three-lines, the first two columns are identical; the first is always HPE\_CUTOFF and the second represents the name of particle type. The values shown in the columns 3-9 define the constants \( r_{\text{min},A(i),A(j)} \) in the energy function of the hydrophobic interactions. Here, \( A(i) \) is the amino acid represented by the current three-lines, and \( A(j) \) is the amino acid represented by columns. The columns 3-9 of the lines 1-3 (totally 21 types) are numbered continuously, and the representing amino acid is the one having the same value of index as defined in the block <<<< para\_cafe\_mol\_lig.

The other additional block is

\[
<<<<\text{cutoff}\_dmax\_hp \\
\text{HPE\_CUTOFF} \quad \text{AA} \quad \text{value} \quad ... \\
(\text{char} \quad \text{char} \quad \text{real} \quad ...) \\
... \\
>>>>
\]

The file format is the same as above, but the \textit{value} defines the constant \( r_{\text{max},A(i),A(j)} \) in the energy function.

10. \texttt{flexible\_local.para}

To be added.

11. \texttt{ion.para}

To be added.

### 4.1.5 AICG files

When the AICG1 model is used together with the option \( i\texttt{\_aicg=2} \) (Note that this is an advanced use of AICG), the AICG1 parameter file has to be prepared as input. The name of the file is defined in the input block “<<<< aicg”. It should have the following format
CHAPTER 4. FILES AND DATA FORMAT

FORMAT(2I10, F10.3) ibd, iunit, K_{b,ibd}(ibd)

: (ibd runs over all the bonds with AICG1 interactions, see Chapter 5)
FORMAT(2I10, F10.3) iba, iunit, K_{θ,iba}(iba)

: (iba runs over all the bond angles with AICG1 interactions, see Chapter 5)
FORMAT(2I10, 2F10.3) idih, iunit, K_{φ, idih}(idih), K_{θ, idih}(idih)

: (idih runs over all the dihedral angles with AICG1 interactions, see Chapter 5)
FORMAT(3I10, F10.3) icon, iunit, just, ε_{go,ij}(icon)

: (icon runs over all the contacts with AICG1 interactions, see Chapter 5)
ibd : index of bonds
iba : index of bond angles
idih : index of dihedral angles
icon : index of native contacts
iunit, junit: index of unit
Note that the order of the above parameters should be consistent with the definition in Cafemol.

Similarly, when the AICG2 model is used with the option i\_aicg=2 (Note that this is an advanced use of AICG2), the AICG2 parameter file has to be prepared as input. The name of the file is defined in the input block “<<<< aicg”. It should have the following format

FORMAT(2I10, F10.3) ibd, iunit, K_{b,ibd}(ibd)

: (ibd runs over all the bonds with AICG2 interactions, see Chapter 5)
FORMAT(2I10, 2F10.3) iba, iunit, ε_{loc,ij}(iba), W_{ij}(iba)

: (iba runs over all the bond angles with AICG2 interactions, see Chapter 5)
FORMAT(2I10, 2F10.3) idih, iunit, ε_{loc,ij}(idih), W_{ij}(idih)

: (idih runs over all the dihedral angles with AICG2 interactions, see Chapter 5)
FORMAT(3I10, F10.3) icon, iunit, just, ε_{go,ij}(icon)

: (icon runs over all the contacts with AICG2 interactions, see Chapter 5)
ibd : index of bonds
iba : the index of bond angle composed of residues i-j (j = i+2)
idih : the index of dihedral angle composed of residues i-j (j = i+3)
icon : index of native contacts
iunit, junit: index of unit
Note that the order of the above parameters should be consistent with the definition in Cafemol.

4.2 Output files

4.2.1 PDB file (.pdb)

The output PDB file has the same format as input PDB file. Please access http://www.wwpdb.org/docs.html for the description of this format.

4.2.2 CARD coordinate (.crd) and velocity (.velo)

The CARD expanded format (also used by CHARMM(Brooks et al., 1983)) is employed to store the final frame of coordinate or velocity, which can be used as restart files. Also, this file is readable from VMD(Humphrey et al., 1996). It is ASCII-type and fixed-column-width, described as below:
4.2 OUTPUT FILES

<table>
<thead>
<tr>
<th>title</th>
<th>natom</th>
<th>EXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i10, 2x, a3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

atomno  resno  res  type  x  y  z  segid  resid  imodel  ctype  iunit  weight

(i10, i10, 2x, a8, 2x, a8, 3f20.10, 2x, a8, 2x, a8, 2xi3.3, a1, i4.4, f20.10

---

Explain:

1. **title** is a title for the coordinate (or velocity). It is a series of lines (maximum 32) of text, with maximum 80 characters per line, each starting with an asterisk (“*”). The title is terminated by a line with the asterisk the first character.

2. Next to **title** is the number of atoms (**natom**), a 10-column-width integer (i10), followed by two spaces (denoted by 2x) and then the string “EXT”.

3. There is one line for each atom.

   - **atomno** gives the index of the atom in the file. It is counted from 1 to **natom**.
   - **resno** gives the residue number of the atom. It is counted from 1 to the total number of residues.
   - **res** gives the residue type of the atom.
   - **type** gives the IUPAC name of the atom.
   - **x**, **y**, and **z** are the coordinate (or velocity) of the atom.
   - **segid** is a string of up to four characters uniquely designating a segment.
   - **resid** is a string of four (or less) characters which uniquely specifies the residue within the segment. The value is counted independently for each segment.
   - **imodel** is the model number of the coordinate.
   - **ctype** is the type of the particle; “P” for protein, “D” DNA, “L” lipid, “R” RNA, and “G” explicit ligand.
   - **iunit** is the unit number of the coordinate.
   - **weight** gives the information of atomic mass.

4.2.3 DCD coordinate (**.dcd**) and velocity (**.vdcd**)

*CafeMol* adopts DCD format (also used by CHARM(Brooks et al., 1983)) and NAMD(Phillips et al., 2005). This format is readable by VMD(Humphrey et al., 1996) so that one can watch the simulation movie to save trajectory in binary. (GNU compiler is not supported.) The format is described as following:
### Files and Data Format

**Table:**

<table>
<thead>
<tr>
<th>blocksize1</th>
<th>hdr</th>
<th>nset</th>
<th>istrt</th>
<th>nsavc</th>
<th>nstep</th>
<th>null</th>
</tr>
</thead>
<tbody>
<tr>
<td>(int)</td>
<td>char+4</td>
<td>int</td>
<td>int</td>
<td>int</td>
<td>int</td>
<td>int+4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>natom − nfreat</th>
<th>delta</th>
<th>null</th>
<th>version</th>
<th>blocksize1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(int)</td>
<td>real</td>
<td>int</td>
<td>int+9</td>
<td>int</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>blocksize2</th>
<th>ntitle</th>
<th>title</th>
<th>blocksize2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(int)</td>
<td>int</td>
<td>char(80)*ntitle</td>
<td>int</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>blocksize3</th>
<th>natom</th>
<th>blocksize3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(int)</td>
<td>int</td>
<td>int</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>blocksize4</th>
<th>x</th>
<th>blocksize4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(int)</td>
<td>real+natom</td>
<td>int</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>blocksize5</th>
<th>y</th>
<th>blocksize5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(int)</td>
<td>real+natom</td>
<td>int</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>blocksize6</th>
<th>z</th>
<th>blocksize6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(int)</td>
<td>real+natom</td>
<td>int</td>
</tr>
</tbody>
</table>

---

**Explain:**

1. There are six types of blocks in the whole file starting and ending with `blocksize` for each, containing simulation information, the title, the total number of atoms, and x-, y-, and z-coordinate (or velocity), respectively. The `blocksize` means the total bytes for a particular block. For example, the size of the block 1 (`blocksize1`) must always be 84.

2. The first block describes parameters of simulation and it contains 8 values plus 13 trivial zeros.
   - `hdr` is the header of the file, and it can be either “CORD” or “VELD”, for coordinate or velocity, respectively.
   - `nset` means the total number of frames in the trajectory.
   - `istrt` indicates the starting time-step. In most cases it is equal to 1, but it may also be larger than 1 when a continuous stimulation is performed for a previous one.
   - `nsavc` is the frequency to save coordinate (or velocity) in the simulation.
   - `nstep` is the total number of time-steps of the simulation.
   - `null` means these positions are reserved. In this case it is an array of four integer zeros.
   - `natom − nfreat` means the number of free atoms (corresponding to fixed atoms in some cases). Currently it is set to be zero.
   - `delta` saves the step-size of the simulation.
   - `null` is an array of nine integer zeros.
   - `version` is set to be 24, which means we pretend to be a CHARMM24 format.

3. The second block describes the title.
   - `ntitle` is the number of title lines, and it should be less than 33.
   - `title` is the real content of the title, which should be managed into single or multiple lines with each line containing exactly 80 characters.

4. The third block saves simply the total number of atoms (`natom`) in the trajectory.

5. Block No. 4-6 save the coordinate (or velocity).
   - `x`, `y`, and `z` are coordinate (or velocity) with respect to x-, y-, and z-axis, respectively.

Note that block No. 3-6 are repeated for a trajectory.
4.2. OUTPUT FILES

4.2.4 PSF (protein structure file) file (.psf)

The PSF file is helpful to visualize structures and to watch movie using VMD (Humphrey et al., 1996). Although the DCD file alone can draw the structure in VMD, one can see “bond” by using PSF file concomitantly.

An example usage of VMD on Linux is as below.

$ vmd -psf filename.psf -dcd filename.dcd

The PSF file format is described at the NAMD website (http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-unix-html/node21.html), though CafeMol outputs only ATOM and BOND information.

4.2.5 Movie file (.movie)

The “movie” format is employed by CafeMol for the storage of trajectory (coordinate) in text. The movie format is in principle resembles the PDB format, except that the coordinates are embedded between “<<<<” and “>>>>”:

\[
<<<< istep tempk (i8, f10.3) \\
Coordinate \\
..... \\
>>>> \\
END
\]

where \( \text{istep} \) is the current time-step, and \( \text{tempk} \) is the temperature. The coordinates are collected with respect to unit, and for each unit the data are embedded in a lower level block:

\[
<< \text{protein\_iunit} (i3) \\
Coordinate \\
..... \\
>>
\]

where \( iunit \) represents the unit number.

4.2.6 Data file (.data)

CafeMol saves the major log of a simulation in the data file. It has a mixture of data format, recording all the setup parameters that are specified in other files.
4.2.7 Time-series file (.ts)

*CafeMol* outputs energy regularly (with a certain frequency specified in the input file) into the time-series file. In this file, the lines starting with “#” are comments (so that it is skipped in gnuplot), and the others are time-series of some basic quantities, of which formats depend on simulations performed. It is a plain text format.

There are two styles of energy output, which can be set in the input file (see Chapter 5):

- \texttt{i\_output\_energy\_style} = 0: the energy terms for each unit are displayed, which include intra-unit energy and half of the inter-unit energy.
- \texttt{i\_output\_energy\_style} = 1: the inter-unit energy terms are subtracted from intra-unit terms and are displayed explicitly.

For a simple simulation of Go model with \texttt{i\_output\_energy\_style} = 0, the output looks like:

```
# initial_energy
# total_energy = etot (f15.3)
# t_series

# ******************************************************************************
# step tempk radg etot velet qscore rmsd
# ******************************************************************************
# all step tempk radg etot velet qscore rmsd local go repul
# ******************************************************************************

# step tempk radg etot velet qscore rmsd
( a5, 1x10, 2(1xf8.2), 2(1xf10.2), 1xf6.3, 1xf8.2)
(a5, 1x10, 2(1xf8.2), 2(1xf10.2), 1xf6.3, 1xf8.2, 3(1xf10.2))

#iunit step tempk radg etot velet qscore rmsd
( a1,i1,3x, 1xi10, 2(1xf8.2), 2(1xf10.2), 1xf6.3, 1xf8.2, 3(1xf10.2))
```

There is one non-comment line for each time-step reporting the results of some order parameters and total energy terms of the whole system. For this line, variables are explained as following: \texttt{step} is the time-step, \texttt{tempk} is the temperature, \texttt{radg} is the radius of gyration, \texttt{etot} is the total energy, \texttt{velet} is the kinetic energy, \texttt{qscore} is the fraction of native contacts, and \texttt{rmsd} is the root-mean square deviation.

The following comment lines report the information for each unit (starts with “#iunit”, where \texttt{iunit} is the unit number), as well as its sum (starts with “#all”). In the case of \texttt{i\_output\_energy\_style} = 1, additional rows starting with “#iunit − junit” in the format “2(a1,i1,3x)”, and corresponding energy terms for the inter-unit interaction are outputed. In addition to the columns described above, \texttt{local} is the energy of local interaction, \texttt{go} is the energy of non-local Go interaction, and \texttt{repul} is the energy of repulsive interaction. Moreover, when “bridge”, “pulling”, “anchor”, “rest1d”, or “in_box” options are turned on (See Chapter 5), the corresponding energy terms for these will be printed as additional columns (It is noted that \texttt{etot} in the non-comment line includes these energy terms).

More energy terms are shown as additional columns when particular interaction are turned. For example, the energy of hydrophobic interaction is displayed as “hp”, electrostatic interaction “elect”, Morse potential “morse”, solvation term of DNA model “solv\_dna”, DNA/RNA stacking “stack”, DNA base-pairing “base”, RNA base-pairing “pair”, and the implicit ligand model “imp\_lig”. In the case of implicit ligand model, additional line is printed, too, at the end of the output of each step:

```
##implicit_ligand_isite (step, energy, state) = ( step, chbond, istate)
(a18, i1, a24, i10, 1x, f10.3, 1x, i1)
...```
where \textit{isite} is the binding site number, \textit{ebind} is the binding energy of ligand and protein, and \textit{istate} indicates the binding state of ligand (=1 bound, =0 unbound).

For replica-exchange simulations, additional “label” column is shown, which indicates the current Hamiltonian label of the replica.

For the case that the system contains multiple-basin Go model, the file looks like (\textit{i_output_energy_style} = 0):

\begin{verbatim}
# initial_energy
# total_energy = etot (f15.3)
# t_series

#---------------------------
# # unit step tempk radg etot velet 1sys
#---------------------------

step tempk radg etot velet 1sys
a5 1x10, 2(1xf8.2), 2(1xf10.2), 1xi3, a3,

#all
step tempk radg etot velet 0.0

(a5, 1x10, 2(1xf8.2), 2(1xf10.2), velet 1xf6.3,

#unit istate
step tempk radg etot velet qscore
(a1,i1,a1,2x, 1xi10, 2(1xf8.2), 2(1xf10.2), 1xf6.3,

... ch_{a-b} ... coef_{a} qscore_{a} estate_{a} coef_{b} qscore_{b} estate_{b} ...

rmse local go repul
ch_{a-b} ... coef_{a} qscore_{a} estate_{a} coef_{b} qscore_{b} estate_{b} ...

1xf6.3, ... 2(1xf6.3), 1xf10.2, 2(1xf10.2), 1xf10.2, ...

0.0 0.0 0.0 repul
1xf8.2 3(1xf10.2) }

rmse local go repul
1xf8.2 3(1xf10.2) }

...
\end{verbatim}

The uncommented line is the summary data for each time-step. The columns from \textit{step} to \textit{velet} describe the whole system in the same way as those in simple Go case. The following columns in this line describe data for each multiple-basin (MB) system (System numbers are indicated by digits and state numbers by lowercase alphabets). Several columns following “1sys”, for example, describe results for the MB system 1, and the columns following “2sys” describe results for the MB system 2, and so on. Within the description of each MB system, the output for each state are also shown in order. \(ch_{a-b}\) is the order parameter \(\chi\) defined between states \(a\) and \(b\). For system of more than two basins, the number of \(ch\) is larger than one, and they are output sequentially in this file. Then, \(coef_{a}\), \(qscore_{a}\), and \(estate_{a}\) are the force coefficient, \(Q\)-score (the fraction of native contact), and energy, respectively, for state \(a\).

Then, in the following comment lines, as in the case of simple Go simulation, the summation of each term over units (starts with “\#all”), as well as the value of term for each state of each unit (starts with “\#unitistate”, where \textit{iunit} and \textit{istate} are the unit number and state number, respectively) are printed. All variables have the same meaning as the case of simple Go simulation explained above. The comments described above on \textit{i_output_energy_style} = 1, as well as the additional outputs for special uses, are also applied here.
4.2.8 Native-info file (.ninfo)

The output native-info file is essentially the same format to the input native-info file. Please see above for a reference of such file format.

4.2.9 Replica information file (.rep)

If you employ the replica exchange method (REM, i.e. “i.run_mode=6” in “<<job_cntl” block), a replica information file (.rep) is automatically generated as an output. This file contains data related to REM such as distribution of variables, time-series and some statistics of replica itinerary.

The first block lists replica-specifying parameters with the assigned ID (setID). (Here, the “set” means combination of replica-specifying parameters, $\Lambda_m$ in Chapter 3.4.7.) If the temperature REM is performed, $\text{replica variable } 1 \Lambda^{[1]}$ corresponds to temperature alone. For 2D replica, the list shows two replica-specifying parameters per line.

<table>
<thead>
<tr>
<th>#table of replica variables and setID</th>
</tr>
</thead>
<tbody>
<tr>
<td>#setID</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>$m$</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>$N_{rep}$</td>
</tr>
<tr>
<td>(i5, 1x,</td>
</tr>
</tbody>
</table>

Next, MD steps between replica exchange attempts are listed for every replicas. This block is informative only when you employ replica-dependent steps for better load-balancing (currently not available).

<table>
<thead>
<tr>
<th># number of exchange steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>$N_{rep}$</td>
</tr>
<tr>
<td>(i5, 1x,</td>
</tr>
</tbody>
</table>

The third block, normally the largest content, is time-series of replica itinerary. The left-most column is the time-step and the other columns show, in order, the setIDs $m(i)$ for $i$-th replica.

<table>
<thead>
<tr>
<th># history of replica exchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>step</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>(i10, 1x,</td>
</tr>
</tbody>
</table>
The next block shows the histogram of replica existences for each setID. Each row (LABEL) indicates the setID \( m \) (temperature ID or ID that specifies the Hamiltonian parameter), and each column \( i \) (REPLICA) corresponds to one consecutive conformational sampling.

\[
\begin{array}{cccccc}
\text{LABEL} & \text{REP00001} & \text{REP00002} & \cdots & \text{REP}(i) & \cdots & \text{REP00024} \\
1 & n(1,1) & n(1,2) & \cdots & n(1,i) & \cdots & n(1,N_{\text{rep}}) \\
\vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
m & n(m,1) & n(m,2) & \cdots & n(m,i) & \cdots & n(m,N_{\text{rep}}) \\
\vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
N_{\text{rep}} & n(N_{\text{rep}},1) & n(N_{\text{rep}},2) & \cdots & n(N_{\text{rep}},i) & \cdots & n(N_{\text{rep}},N_{\text{rep}}) \\
(j6,1x,) & i8,1x, & i8,1x, & \cdots & i8,1x, & \cdots & i8)
\end{array}
\]

The last block shows acceptance ratio of exchange for each neighboring setID pairs.

\[
\begin{array}{cccccc}
\text{LABEL-LABEL ratio} & & & & & \\
1 & 2 & \text{ratio}(1,2) & \\
2 & 3 & \text{ratio}(2,3) & \\
\vdots & \vdots & \vdots & \\
j & k & \text{ratio}(j,k) & \\
\vdots & \vdots & \vdots & \\
(N_{\text{rep}} - 1) & N_{\text{rep}} & \text{ratio}(N_{\text{rep}} - 1,1) & \\
(2x,5,a1, & i5,2x, & f10.8)
\end{array}
\]
Chapter 5

Input file: How to make

5.1 General rules of input files

First of all, we expect (hope!) that CafeMol runs appropriately only when an input file is prepared consistently throughout the file. When you use an inconsistent input file, CafeMol is not necessarily kind enough to stop with an error message, but quite often still runs and provides some results, but those results are most likely meaningless. It is your responsibility to make a consistent input file!. You should read through the manual before doing simulations.

The input file of CafeMol uses general rules. Knowledge of them will be essential to write and understand input files.

1. Each of the input lines has to be equal to or less than 256 (this number is defined by CARRAY_MXCOLM in const_maxfile.F90 ans thus is changeable) columns. CafeMol reads up to 256 columns and just ignores anything written in columns after 256, without error message.

2. The input file consists of many “input blocks”. Each input block has the style,

```
<<<< name_of_input_block
contents of input block that contain many lines
>>>>
```

Namely, each input block starts from “<<<<” that should be from the 1st column to the 4th column. This is followed by the name_of_input_block that specifies the input block. Each input block ends with “>>>>” that also should be from the 1st column to the 4th column. One can comment out the input block by comment out the initial line as

```
*<<<< name_of_input_block
contents of input block that contain many lines
>>>>
```

The above input block is entirely ignored.

For every CafeMol input files, several input blocks are required, and others are optional. Orders of these input blocks are not essential. You can order them freely.

The input file is completely made of input blocks. Any message out of blocks are just ignored.

Within an input block, many lines are to be written. Orders of lines are almost free (There are some exceptions that are described in each section).

3. Lines that have “*” in the 1st column are taken as comments. Blank lines are also ignored.
4. There are only two styles of lines in the input file, each of which will be described in 5 and 6.

5. Many lines in input files contain the style,

\[ \text{name_of_parameter} = \text{value} \]

For this, the style is quite flexible. \text{name_of_parameter} can start from any column. Between \text{name_of_parameter} and \text{=}, and between \text{=} and \text{value}, any length of space (including zero) is allowed. (There is an exception that the left-hand side is value and the right-hand-side is a file name in “<<<< native_info_simN”)

6. Many lines in input files start from the UPPERCASE words.

\[ \text{PARAMETER} \ldots \ldots \]

For this uppercase word, one has to write it from the 1st column. On these lines, usually up to the 1st space, the format is quite rigid.

5.2 “<<<< filenames (required)

This input block sets paths, input and output filenames, and output data formats.

The following line defines the directory where output data are saved.

\[ \text{path} = \text{directory} \quad \text{(default “./”)} \]

The next line sets the output file names up to the suffix.

\[ \text{filename} = \text{name_of_file} \quad \text{(required)} \]

The following specifies what types of output files you are requesting to get.

\[ \text{OUTPUT} \quad \text{names_of formatter_to_be_output} \quad \text{(optional)} \]

Out of several available formats listed below, you can write any numbers of the formats. The available formats of trajectory output are pdb, crd, velo, movie, dcd, and vdcd. The former 3 save data only several times (this number is defined by \text{I\_RECORD} in \text{const\_maxsize.F90} and thus is changeable) in a run and are used primarily for a quick view of overall structural changesrecord. On the other hand, the rest save data much more frequently and can be used for trajectory analysis and for making movie. On top, two other formats, psf and rst, are also available. The psf (PSF; Protein Structure File) is useful to visualize trajectories by VMD. (See 4.2.4) and the rst format is needed to restart simulation continuously. (See 2.3)

1. pdb; PDB file format for (coordinate)

2. crd; CARD file format for (coordinate). This allows one to save longer digits coordinates than the PDB format. For very large systems, this format is required.

3. velo; CARD file format for (velocity).
4. movie; PDB file format for trajectory (coordinate)

5. dcd; DCD file format for trajectory (coordinate), which can be put into VMD for watching movie. *(CafeMol compiled by GNU compiler can not run with this option.)*

6. vdcd; DCD file format for trajectory (velocity), which may not be used very often. *(CafeMol compiled by GNU compiler can not run with this option.)*

7. psf; PSF format for better drawing with VMD

8. rst; restart file

The next line defines the directory where the parameter files are prepared.

```
path_para = directory (default “./para”)
```

The next line defines the directory where the native (reference) structures are prepared.

```
path_pdb = directory (default “./pdb”)
```

The next line defines the directory where the initial structures are prepared.

```
path_ini = directory (default “./pdb”)
```

The following defines the directory where the native (reference) information is prepared.

```
path_natinfo = directory (default “./ninfo”)
```

The following “path_aicg” sets the directory of files related to the AICG model, including the dssp file.

```
path_aicg = directory (default “./aicg”)
```

An example of this input block is

```
<<<< filenames
path = ./data
filename = cafemol_mgo_1chain
OUTPUT crd velo dcd pdb rst
path_para = ./para
path_pdb = ./pdb
path_ini = ./pdb
path_natinfo = ./ninfo
>>>>>
```
CHAPTER 5. INPUT FILE: HOW TO MAKE

5.3  job_cntl (required)

This input block sets the overall job that one wants to do. This block only specifies 4 parameters, of which the first three are required to be specified.

This defines the basic run mode and can take one of the listed values

```
<table>
<thead>
<tr>
<th>i_run_mode</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Debug mode. Check the consistence between force and energy</td>
</tr>
<tr>
<td>2</td>
<td>Constant temperature simulation</td>
</tr>
<tr>
<td>3</td>
<td>Simulated annealing (require &quot;annealing&quot; field)</td>
</tr>
<tr>
<td>4</td>
<td>Auto-search of $T_F$ (require &quot;searching_tf&quot; field)</td>
</tr>
<tr>
<td>5</td>
<td>Energy calculation at single point (not implemented yet)</td>
</tr>
<tr>
<td>6</td>
<td>Replica exchange simulation</td>
</tr>
<tr>
<td></td>
<td>(require &quot;replica&quot; and &quot;replica_XXX&quot; fields)</td>
</tr>
</tbody>
</table>
```

This defines type of dynamics and can take one of the following values

```
<table>
<thead>
<tr>
<th>i_simulate_type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Newtonian dynamics (velocity Verlet)</td>
</tr>
<tr>
<td></td>
<td>with the constant energy (not working)</td>
</tr>
<tr>
<td>1</td>
<td>Langevin dynamics (recommended)</td>
</tr>
<tr>
<td>2</td>
<td>Newtonian dynamics (velocity Verlet)</td>
</tr>
<tr>
<td></td>
<td>with Berendsen thermostat</td>
</tr>
<tr>
<td>3</td>
<td>Newtonian dynamics (velocity Verlet)</td>
</tr>
<tr>
<td></td>
<td>with Nose-Hoover thermostat</td>
</tr>
<tr>
<td>4</td>
<td>MPC dynamics</td>
</tr>
<tr>
<td></td>
<td>(require &quot;mpc_dynamics&quot; (see section 5.35))</td>
</tr>
</tbody>
</table>
```

The following defines the initial conformation of molecules.

```
<table>
<thead>
<tr>
<th>i_initial_state</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Random configuration</td>
</tr>
<tr>
<td>2</td>
<td>Native configuration</td>
</tr>
<tr>
<td>3</td>
<td>Configuration given in the input</td>
</tr>
<tr>
<td></td>
<td>(require &quot;initial_structure&quot;)</td>
</tr>
<tr>
<td>4</td>
<td>B-type DNA configuration</td>
</tr>
<tr>
<td>5</td>
<td>Rectangle lipid-sheet configuration (not yet released)</td>
</tr>
<tr>
<td>6</td>
<td>Configuration given in the input with the CafeMol (CG) style</td>
</tr>
<tr>
<td>7</td>
<td>Configuration given in CARD-style file</td>
</tr>
<tr>
<td></td>
<td>(require &quot;initial_structure&quot;)</td>
</tr>
</tbody>
</table>
```

The following defines the initial velocities of molecules.

```
<table>
<thead>
<tr>
<th>i_initial_velo</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Random velocity drawn from the Maxwell-Boltzmann distribution (default)</td>
</tr>
<tr>
<td>1</td>
<td>Velocity given in CARD-style file</td>
</tr>
<tr>
<td></td>
<td>(require &quot;initial_velo&quot;)</td>
</tr>
</tbody>
</table>
```

A typical example is
which sets the Langevin dynamics at a constant temperature starting from a conformation given in the input file.

When “i_initial_state=2” and the multiple-Go model is used, the native configuration is chosen from the state a.

### 5.4 <\texttt{unit_and_state} (required)

This input block defines molecular units (nearly same as chains) and states of the simulated molecular system. On top, this input block specifies the way to input sequence and native-structural information of molecules.

First, there are basically 2 ways to input sequence information, either from pdb files or directly from the input file using “<\texttt{sequence} block”. In either way, the format for providing sequence information is not different, the use of SEQRES lines of pdb styles. The way to specify the sequence information is given as

\begin{verbatim}
    i_seq_read_style = integer
    (required) 1 : from PDB files (The pdb file names need to be specified below in this input block )
    2 : directly from the input file (This requires “<\texttt{sequence} block)
    3 : from initial_lipid block (not released)
    4 : from CafeMol (CG) style (The pdb file names need to be specified below in this input block)
\end{verbatim}

Second, there are basically 3 ways to input native-structure information used for Go model or its derivatives.

\begin{verbatim}
    i_go_native_read_style = integer
    (required) 1 : from PDB files (The pdb file names need to be specified below in this input block )
    2 : from “native_info” files (This requires “<\texttt{native_info_simN” block, where N=1,2,...)
    3 : none (in case no Go interaction is used at all)
\end{verbatim}

The above defines the way to input native-structure information.

Note that if “i_go_native_read_style = 1”, you must also set “i_seq_read_style =1”.

Currently, i_go_native_read_style = 2 is not compatible with AICG2 model. Namely, when one uses AICG2 model, one cannot provide the native-structure information from “native_info” files.
Third, molecular units and states are to be specified. An "unit" is equal to a chain for proteins and nucleic acids. For ligands and lipids, an unit corresponds to a set of all ligands and a set of all lipid molecules. The units are identified by integers from 1, 2,... A "state" is used in multiple-basin (or multiple Go) model and the state is identified by an alphabet starting from a, b, ... .

The "" unit_and_state" block lists all the units and states of the simulated system together with their molecular type (protein, dna, lipid,or ligand) and the pdb file of the native structures (when used). Each line has the form

```
    u&$ molecular_type PDB_file
```

(required at least 1 line) The "u&$" stands for unit_and_state For the unit that is not modelled by the multiple-Go model, u&$ is just an integer representing its unit (chain) number. For the multiple-Go model, the unit number is followed by an alphabet, a, b, ,,for defining the state.

The molecular_type is either protein, dna, rna, lipid(not available), or ligand. When a pdb file includes more than one molecular_type, you should divide the pdb file into two files, each of which contains one molecular_type.

PDB_file is the name of a PDB file that represents the native (reference) structure. This is required when "i_seq_read_style = 1". Then, the sequence information is read from the pdb file. On top, when "i_go_native_read_style = 1", native-structural information is also read from this pdb file. Here, we emphasize again that, when the native-structural information is provided by pdb files, these pdb files need to contain coordinate information of ALL-ATOMS, but not Ca coordinates alone. When "i_seq_read_style = 2", you should make "" " sequence" block in the input file.

The unit number has to be in ascending order from the top to the bottom lines.

Examples should help one’s understanding. For a single protein that is modelled by the Go model,

```
<<<< unit_and_state
i_seq_read_style = 1
i_go_native_read_style = 1
1 protein hoge.pdb
>>>>>
```

is fine. For a single protein that is modelled by a double-Go model,

```
<<<< unit_and_state
i_seq_read_style = 1
i_go_native_read_style = 1
1a protein hoge1.pdb
1b protein hoge2.pdb
>>>>>
```

When the simulated system contains more than one unit, and you have a pdb file that contains native structures of more than one units, one can use this PDB file to define unit_and_state of more than one molecules at once. An example is
which sets 3 units (the units 2, 3, and 4) in the state a simultaneously. Assumed here is that “hoge234.pdb” contains the native structure of the 3 units. Alternatively, one can provide one pdb file for one unit, too.

5.5  <<<<< native_info_simN (N can be 1, 2,,,) (optional)

This input block is required when “i_go native read style=2” in “<<<< unit and state” block.

For using Go model and its relatives, CafeMol requires input of the native (or reference) structure information of proteins anr/or nucleic acids. In many simple cases, the native structure information is provided by the PDB file listed in the “<<<< unit and state” block (This corresponds to “i_go native read style=1” case). However, CafeMol equips alternative way described here, which has much more flexibility. This input block is to define the native structural information of Go model of proteins in a more flexible way.

Importantly, when you “switch” the native (reference) structure during the simulation, you MUST USE this “native_info style” for defining the native structure.

Currently, when one uses AICG2 model, one cannot use this style.

We also note that, with this input style, you can create a Go model that is not biased to one structure, but that contains some FRUSTRATION. For example, we can set the native contacts in the way that some native contact information taken from structure 1 and other native contact information taken from structure 2, resulting in frustrated interactions.

Within the “native_info_simN” specification of the native structure information, there are two styles of writing in this input block. 1) All-in-one-file style: Define native (reference) structural information of all the units in a single file, and 2) One-by-one-file style: Define the native information of each unit in a separate file. The former is easier because one can copy&paste, and edit if necessary, the output data stored in the native-info file (.ninfo) in a previous run. The latter is sometime more flexible.

The format for the native-info file is described in Chapter 4. We remark that the usage of the data is slightly, but importantly, different between the two styles.

5.5.1 All-in-one-file style

By this, native structural information of all the units are given in a file. In the input file, all you need is to define the filename of it.

\[
\text{NINFO(all/all) filename}
\]

where filename specifies the file name that contains all the native structural information.

In this scheme, the columns of (absolute) sequential numbers of mp’s (imp1, imp2,,,) in the native-info file are read (See Chapter 4).

5.5.2 One-by-one-file style

In this scheme, contents of the block are made of two types of lines. The first is

\[
\text{NINFO(u&s1/u&s2) integer}
\]
which states that native structural information of the interaction between the unit\&state \( u_s \) and the unit\&state \( u_s \) is defined in a file identified with an \textit{integer}. For systems with many units, many lines of this type are to be written. These particular integer numbers are then linked to particular files in the second type of lines as

\[ \text{integer} = \text{filename} \]

where the \textit{integer} here should be equal to the integer found in lines of the first type. The \textit{filename} is the name of the file that contains native structural information.

We note that, in this scheme, the columns of sequential numbers of mp in each unit (e.g., imp1un, imp2un,\ldots) in the native-info file are read. This style would be very useful to dynamically permuted the native-structural information in the simulated complex, such as the case of F\(_1\)-ATPase.

### 5.6 \textless\textless\textless initial\_struct (optional)

This input block specifies the initial structures of all units. This block is optional, but when “i\_initial\_state” in the “job\_cntl” block is set as 3 (actually this is the case for most applications), this block is required.

The “initial\_struct” block lists all the units and the corresponding PDB files that contain the initial conformation in the form,

\[
\text{unit PDB\_file\_for\_initial\_conformation} \quad \text{(required)}
\]

The unit number has to be in the ascending order from the top to the bottom lines.

Examples are

\begin{verbatim}
\textless\textless\textless initial\_struct
1 hoge1.pdb
2 hoge2.pdb
\textgreater\textgreater\textgreater
\end{verbatim}

Note that one should not put the state id, a or b after the unit number even if you are using multiple-Go model for some units; The initial structure should be unique! As before, \textit{CafeMol} accepts the assignment of more than one unit simultaneously, like the form

\begin{verbatim}
\textless\textless\textless initial\_struct
1-2 hoge12.pdb
\textgreater\textgreater\textgreater
\end{verbatim}

Of course, “hoge12.pdb” should contain the initial structures of the units 1 and 2.
5.7 <<<<< initial_velo (optional)

This input block specifies the initial velocities of all units. This block is optional, but when “i_initial_velo” in the “jobcntl” block is set as 1, this block is required.

The “initial_velo” block lists all the units and the corresponding CARD-style files that contain the initial velocities in the form,

\[
\text{unit } \text{PDB file for initial velocity} \text{ (required)}
\]

The unit number has to be in the ascending order from the top to the bottom lines.

Examples are

\[
<<<< initial_velo
1 hoge1.velo
2 hoge2.velo
>>>>
\]

Note that one should not put the state id, a or b after the unit number even if you are using multiple-Go model for some units; The initial velocity should be unique! As before, CafeMol accepts the assignment of more than one unit simultaneously, like the form

\[
<<<< initial_velo
1-2 hoge12.velo
>>>>
\]

Of course, “hoge12.velo” should contain the initial velocities of the units 1 and 2.

5.8 <<<<< initial_lipid (not implemented)

5.9 <<<<< sequence (optional)

This input block is required only when “i_seq_read_style=2” in “<<<< unit_and_state” block. In this case, no PDB file is specified at the 3rd column of the input block “<<<< unit_and_state”. Both protein and nucleic acid sequences can be given. SEQRES style in the PDB format (note that this format is very rigid) is used.

5.10 <<<<< dssp_file (optional)

This input block is required only when the AICG1 model is used together with the option i_aicg=1. The “<<<< dssp_file” block lists all the units and the corresponding dssp files that contain the needed secondary structure information.
The dssp files have to be specified with the same order as the pdb files in the “<<<<<< unit_and_state” block, such that the dssp files and pdb files have one-to-one correspondence. Before running CafeMol, one needs to prepare the corresponding DSSP result files at the directory specified as the path_aicg in the input block “<<<<<< filenames”.

The required contents of the dssp files are exactly the output files of the DSSP program. So, to use this option, beforehand, one needs to run the DSSP program and copy-and-paste the resulting output file here. The DSSP program is available from its homepage “http://swift.cmbi.ru.nl/gv/dssp”. For each pdb file, the secondary structure is calculated independently, and the resulting dssp files should be put into the directory “path_aicg” specified in the input block “<<<<<< filenames”.

5.11 <<<<<< energy_function (required)

This input block sets the energy function used for each of simulating units. This block consists of a few types of lines.

First, one defines types of local and non-local interactions you want to use. For the local interaction, one specifies it for each unit and state (the first line below). For the non-local interaction, one specifies it for each pair of unit and states (the second line below). The formats are

\[
\begin{align*}
\text{LOCAL}(u&s_1) & \text{ local_force_type} \\
\text{NLOCAL}(u&s_1/u&s_2) & \text{ non_local_force_type, (nonlocal_force_type,...)}
\end{align*}
\]

Each unit and state is represented by an integer (followed by an alphabet when multiple-Go model is used). For the local interaction, one has to specify one local_force_type. On the other hand, one can specify one or more than one non_local_force_type. For representing nonlocal INTRA-chain interaction, one specifies the same unit_and_state before and after the slash /.. Candidates for local_force_type are,

<table>
<thead>
<tr>
<th>local_force_type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOTHING</td>
<td>no interaction</td>
</tr>
<tr>
<td>L_GO</td>
<td>local Go potential (bond, bond angle, dihedral angle)</td>
</tr>
<tr>
<td>L_AICG1</td>
<td>local AICG1</td>
</tr>
<tr>
<td>L_AICG2</td>
<td>local AICG2 (including flexible local potential)</td>
</tr>
<tr>
<td>L_FLP</td>
<td>flexible local potential used for the entire unit</td>
</tr>
<tr>
<td>L_BOND</td>
<td>bond potential only</td>
</tr>
<tr>
<td>L_ENM</td>
<td>elastic network model(=NOTHING)</td>
</tr>
<tr>
<td>L_BDNA</td>
<td>local DNA interaction</td>
</tr>
<tr>
<td>L_RIGID_LIG</td>
<td>ligand rigid interaction</td>
</tr>
</tbody>
</table>

We note that “L_FLP” is to be chosen only when one wants to use the flexible local potential for the entire unit. Otherwise when one wants to use the flexible local potential for some portions of an unit, this will be treated by switchin on “i_flp=1” (see below).

Candidates for non_local_force_type are,
5.11. **ENERGY_FUNCTION (REQUIRED)**

<table>
<thead>
<tr>
<th>Interaction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOTHING</td>
<td>no interaction (default)</td>
</tr>
<tr>
<td>GO</td>
<td>12-10 Go potential for native-contact pairs, ( \varepsilon_{go} \left[ 5 \left( \frac{r_{ij}}{r_{ij}} \right)^{12} - 6 \left( \frac{r_{ij}}{r_{ij}} \right)^{10} \right] )</td>
</tr>
<tr>
<td>EXV</td>
<td>repulsion, ( \varepsilon_{ev} \left( \frac{d_{ij}}{r_{ij}} \right)^{12} )</td>
</tr>
<tr>
<td>ELE</td>
<td>electrostatic interaction (Debye-Huckel form)</td>
</tr>
<tr>
<td>DNA</td>
<td>DNA-DNA interaction</td>
</tr>
<tr>
<td>ENM</td>
<td>elastic network model</td>
</tr>
<tr>
<td>HP</td>
<td>hydrophobic interaction</td>
</tr>
<tr>
<td>MORSE</td>
<td>Morse Go potential</td>
</tr>
<tr>
<td>PAIR_RNA</td>
<td>RNA basepair</td>
</tr>
<tr>
<td>AICG1</td>
<td>AICG1 contact potential</td>
</tr>
<tr>
<td>AICG2</td>
<td>AICG2 contact potential</td>
</tr>
</tbody>
</table>

For the Go model and the multiple-Go model, use “GO” and “EXV”. The hydrophobic interaction can be assigned for any unit-pair. By default, all the hydrophobic particles in the units are considered for the calculation of hydrophobic term. Here, the hydrophobic residues are defined by positive values of \( \varepsilon_{HP} \), \( \varepsilon_{HP,A(i)} \). However, one can also focus on only partial of the unit for hydrophobic interaction, by explicitly defining them in the “<<< hydrophobic” block.

When “<<< native_info_simN” (described elsewhere) block is used to define the native structural information, the native contact set is explicitly defined by the native_info file and thus “GO” in the current block is not affected. Whether or not “GO” is indicated, the native contact indicated in the native_info file has Go potential.

An example of the current input block is

```
LOCAL(1a) L_GO
LOCAL(1b) L_GO
NLOCAL(1a/1a) GO EXV
NLOCAL(1b/1b) GO EXV
```

which defines the multiple-Go model for the unit 1.

In the same way as above, non-local interactions of more than one units can be defined simultaneously such as the example,

```
LOCAL(1-60) L_AICG2
NLOCAL(1-60/1-60) AICG2 EXV
```

where 60 chains are included and all interactions among 60’s are the AICG2 model.

We note that in the old version of CafeMol, we used numeric value instead of alphabet to specify interactions. For example, “GO” was “2” and “EXV” was 3. This old style still works at the moment, but is not recommended at all.

The multiple-Go model is to construct multiple basins in the potential energy function. This can be done for a part of the simulated system. Thus, in one simulated molecular system, there can be more than one (independent) multiple Go models. Each of the multiple Go models is called “system” and need to be defined independently by interactions involved in the system, where the interaction is specified by the pair of unit_and_states. So, for each “system”, multiple-Go model is defined by a set of lines in the form,
A simple example is

\[
\text{MULTIGO\_SYSTEM(1a) 1a/1a} \\
\text{MULTIGO\_SYSTEM(1b) 1b/1b}
\]

which is used for the two-basin model of a single chain. Note that “1” in the “(1a)” stands for the system number, and “1” in the “1a/1a” is the unit number. When two units and its interaction with other chain is included in the system, the input can take the form such as

\[
\text{MULTIGO\_SYSTEM(1a) 2a/2a 3a/3a 2a/3a 1/2a} \\
\text{MULTIGO\_SYSTEM(1b) 2b/2b 3b/3b 2b/3b 1/2b}
\]

where the unit 1 may correspond to the (single) Go model.

The third content in this input block is some small options. There are, at the moment 4 parameters you can specify (all are optional).

\[
\text{i\_flp = integer} \\
\text{i\_use\_atom\_protein = integer} \\
\text{i\_use\_atom\_dna = integer} \\
\text{i\_go\_atom\_dna = integer}
\]

The first line specifies whether flexible local potential is turned on or not. We note that, when one of units uses AICG2 or L\_FLP as the local potential, the flexible local potential is automatically turned on for the corresponding unit and thus one does not need to specify “i\_flp” here. When one wants to use flexible local potentials only for some portion of a unit, e.g, a loop region, one can turn this switch on in the following way.

\[
i\_flp = 0 : \text{additional flexible local potentials are not added (default).}
= 1 : \text{flexible local potentials are turned on. ("<<< flexible\_local" is required)}
\]

The second and the third lines define which of atoms to be used as the representative for the CG particle. For protein (the second line),

\[
i\_use\_atom\_protein = 0 : \text{for using C\_a atom (default)} \\
= 1 : \text{for using C\_b atom} \\
= 2 : \text{for using the center of mass of sidechain}
\]
and for DNA,

```plaintext
i_use_atom_dna = 0 : for using center of mass of phosphate, sugar, and base (default)
```

The fourth line defines which DNA particles are included for Go potential.

```plaintext
i_go_atom_dna = 0 : for using center of mass of phosphate, sugar, and base
               = 1 : for using center of mass of sugar and base (not using phosphate) (default)
```

Together, an example of the “<<<<.energy_function” block would be

```plaintext
<<<< energy_function
LOCAL(1a) L_GO
LOCAL(1b) L_GO
NLOCAL(1a/1a) GO EXV
NLOCAL(1b/1b) GO EXV
MULTIGO_SYSTEM(1a) 1a/1a
MULTIGO_SYSTEM(1b) 1b/1b
>>>>
```

### 5.12. <<<< flexible_local (optional)

This input block is required in the following three cases: (1) “i_flp = 1” in “<<<<energy_function” block, (2) the local potential of at-least one chain is set to AICG2 in “<<<<energy_function” block, or (3) the local potential of at-least one chain is set to “L_FLP” in the same block. In any cases, you should specify the following two variables.

```plaintext
k_dih = real (required)
k_ang = real (required)
```

These variables are the weights of flexible local potentials for the dihedral angle and the virtual bond angle and are normally set to 1.00.

In addition to these variables, in the case of (1) “i_flp = 1”, you should specify the range of using the flexible local potential, by either of the following two variables.

```plaintext
DEL_LGO_ADD_FLP(aa_{ini}-aa_{las}) (can be more than one line)
FLP_ADD(aa_{ini}-aa_{las}) (can be more than one line)
```

In these variables, \(aa_{ini}\) and \(aa_{las}\) are residue numbers of the initial and final amino acids. In either cases, the flexible local potential will be applied to the potential terms that use coordinate information of at least one residue in this range. In the case of “DEL_LGO_ADD_FLP”, the local Go interaction (in the form of bond angle and dihedral angle for simple Go and AICG1, and in the form of 1-3 and 1-4 Gaussian potentials for AICG2) in the specified region will be removed.

When you want to remove the flexible local potential in some regions, you should specify the following variable. Similar to the above box, flexible local potentials that include at-least one residue within the specified range will all be removed.
5.13  `aicg` (optional)

The input block “aicg” sets the AICG options and the names of the AICG parameter files. When AICG1 or AICG2 was specified as local or nonlocal interactions in the “energy_function”, one needs to specify the i_aicg parameter. Note that the AICG2 are always used in conjunction with the flexible local potential, regardless of the setting on i_flp in “energy_function”. One should not switch i_flp = 1 for the use of AICG2.

\[
i_{\text{aicg}} = \text{integer}
\]

1 : AICG; parameters auto-generated by CafeMol (default)
2 : AICG; parameters given by users

If \(i_{\text{aicg}} = 1\) is assigned, the CafeMol automatically builds the heterogenous AICG1 or AICG2 parameters based on both the atomic details of the residue contacts auto-generated by CafeMol and the secondary structure information provided by the user (in case of AICG1). When AICG1 is used as local or nonlocal energy function (specified in the “energy_function” block) together with the option \(i_{\text{aicg}} = 1\), the file name(s) and the directory name which contains(s) the secondary structure information have to be specified by the user in the input block “dssp_file” and by the input block “filenames”, respectively. For the detail, see the corresponding parts in this chapter. When AICG2 is used together with the option \(i_{\text{aicg}} = 2\), one needs not to prepare the secondary structure information file.

If \(i_{\text{aicg}} = 2\) is assigned, the AICG1 or AICG2 parameters need to be input by the user. This option is designed only for advanced users to realize more flexible simulations. To use this option, the name of the parameter file containing all the needed AICG parameters have to be specified in this block. The directory for these files are defined in “filenames”.

In the case of \(i_{\text{aicg}} = 2\), the AICG parameter file has to be prepared by users and the file name is specified by filename_aicg for AICG1 and filename_aicg2 for AICG2. See “energy_function” block for the assignment of the AICG1 and AICG2 interactions.

\[
\text{filename_aicg} = \text{para_aicg}_1\text{.file} \\
\text{filename_aicg2} = \text{para_aicg}_2\text{.file}
\]

(required when \(i_{\text{aicg}} = 2\))

5.14  `multiple_go` (optional)

This is to specify parameters needed for multiple Go model simulations and thus is required only when “MULTIGO_SYSTEM” lines exist in “energy_function”.

\[
\text{bdemax_mgo} = \text{real} \quad \text{(required)}
\]

which sets the upper limit of the change in bond stretching energy \(\epsilon_{b_{\text{max}}}\) (in “kcal/mol/Å²” unit). A typical value is 100.

\[
\text{baemax_mgo} = \text{real} \quad \text{(required)}
\]
which sets the upper limit of the change in bond angle energy $\epsilon_{\theta,\text{max}}$ (in “kcal/mol/radian$^2$” unit). A typical value is 1.

\[
\text{dihemax}_\text{mgo} = \text{real} \quad \text{(required)}
\]

which sets the upper limit of the change in dihedral angle energy $\epsilon_{\phi,\text{max}}$ (in “kcal/mol” unit). A typical value is 0.5.

\[
\text{ENERGAP}(\text{istep}_\text{sim}) (\text{i}_\text{sys}) \quad \text{real real,} \quad \text{(required)}
\]

which defines, for a step of simulation (indicated by “istep_sim”), and for a multiple-Go system (specified by “i_sys”), the energy shifts of every basins. For $n$-basin potential, you need $n$ real numbers, each of which represents the energy shift of each basin, starting from the state $a$, to the $b$, $c$, ...,

\[
\text{DELTA}(\text{i}_\text{sys} \quad \text{j}_\text{state}} \quad \text{k}_\text{state}) \quad \text{real} \quad \text{(required)}
\]

which defines $\Delta$ of the multiple Go model. The $\Delta$ that connects between the state $j$ and $k$ in the $i$-th multiple-Go system. For every pair of states in every multiple Go systems, you need one line of input. 

<<<<< elastic_network (optional)

This input block is required only when the elastic network model ENM is used.

Here, two parameters in the ENM must be specified.

\[
\begin{align*}
\text{cenm} &= \text{real} \quad \text{(required)} \\
\text{dfcontact}_\text{enm} &= \text{real} \quad \text{(required)}
\end{align*}
\]

where “cenm” is the spring constant in the ENM, and “dfcontact_enm” is the threshold length for defining the spring. Note that the contact definition is all-atom based, as in Go model case.

5.15 <<<<< electrostatic (optional)

This input block is used to change the electrostatic interaction when it is turned on in the “<<<< energy_function” block. Note that the default values for these parameters are given in ”electrostatic.para”.

First, the four parameters that specify the electrostatics are

\[
\begin{align*}
\text{cutoff} &= \text{real} \quad \text{(Å)} \\
\text{ionic_strength} &= \text{real} \quad \text{(in molar/liter, 1M NaCl has its value 1)} \\
\text{diele_water} &= \text{real} \quad \text{(dimension-less ratio, for water at 300K this is about 78)} \\
\text{i_diele} &= 0 \quad \text{(for constant dielectric constant)(no other value)}
\end{align*}
\]

The cutoff length should be chosen in a consistent way as the Debye length of the simulated system.

Second, in the same block, the following format is used to define charges for each type:
CHAPTER 5. INPUT FILE: HOW TO MAKE

<table>
<thead>
<tr>
<th>CHARGE_TYPE</th>
<th>AA charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>(char char real)</td>
<td></td>
</tr>
</tbody>
</table>

One line re-defines the charge for the corresponding type of the particle. One can write more than one lines of this type.

In addition, one can specifically turn on/off charges in specific particles.

<table>
<thead>
<tr>
<th>CHARGE_DEL</th>
<th>u&lt;s u residue1, residue2,...</th>
</tr>
</thead>
<tbody>
<tr>
<td>(required)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHARGE_ADD</th>
<th>u&lt;s u residue1, residue2,...</th>
</tr>
</thead>
</table>

The first line shows residues at which charges are to be deleted. The second line shows residues of which charges are to be added. When it is added, Lys, Arg, and His obtain +1 charges, while Asp and Glu have -1 charge. The character "u" must exist before the aa numbers indicating that the numbering is the intra-unit numbering. One can write any number of lines. We note that CafeMol reads through and treat these changes sequentially from the top line to the bottom. Thus, order of lines is of crucial importance. Always, the change specified in lower line has higher priority than the change in upper line.

5.16 <<<<< hydrophobic (optional)

This block can be used to change the hydrophobic (HP) parameters and particles when the hydrophobic interactions are turned on in “<<<<<<< energy function” block.

First, one can change the following parameters,

- \( \text{coef}_h p \) : the constant \( c_{HP} \) in the energy function of the hydrophobic interaction. (0.58 by default)
- \( \text{coef}_r h o_h p \) : the constant \( c_{\text{linear}} \) in the hydrophobic term. (0.2 by default)
- \( \rho_{\text{min}}_h p \) : the constant \( \rho_{\text{min}} \) in the hydrophobic term. (0.3 by default)

by the form

\[ \text{name of parameter} = \text{value} \]

Note that this has the identical form as the lines in hydrophobic.para where the default values are given.

Second, the strength of hydrophobicity of 21 particle types (20 amino acids plus one non-amino acid) can be changed by the form,

| <<<<<para_cafemol_hp |
| HPE_TYPE | AA epsilon_HP n-j n-j max 1j |
| (char char real real real char int) |

... >>>>>>
Note that these forms are identical to those in hydrophobic.para where the default values are given. Each line defines the parameters for one amino acid or other hydrophobic particles. The first column is the keyword. The second column lists the amino acid name. The third, fourth, and fifth columns define the values for the constants $\epsilon_{HP,A(i)}$, $n_{A(j)}$, and $n_{\text{max},A(i)}$, respectively, in the energy function of the hydrophobic interaction. The last column is a comment, defining the index of each amino acid. The last column will not be read by the program, but is shown just for the convenience of remembering amino acid indices.

When the hydrophobic interactions are turned on for a pair of units in “<<<< energy function”, by default, all the hydrophobic particles in the units have hydrophobic interactions. (Note that the hydrophobic particles are defined by the positive $\epsilon_{HP,U}$ specified in the hydrophobic.para or in the above-mentioned inputs). But, sometime, one may change (most often “restrict”) the hydrophobic particles. The following lines allow one to change the default and generic assignment to the specific assignment. There are two ways of changes; one is to delete some HP sites, and the other is to add some HP sites.

```
HPE_DEL unit u residue1 residue2,,
HPE_ADD unit u residue1 residue2,,
```

(can be more than one line)

Here, the `unit` is the unit number, in the first line, `residue1`, `residue2`, indicate residue numbers of which HP interactions are turned off. In the second line, HP interactions are added. Here, the residue should be numbered in the intra-unit, but not in the accumulated fashion. The character “u” must exist before the residue numbers indicating that the numbering is the intra-unit numbering. One can define the hydrophobic sites for the same unit by multiple lines, each starting with the “HPE_DEL” or “HPE_ADD” and the unit number. The residue numbers are separated by space, and continuous numbers are allowed to be written in concise format. For example, “1 2 3” can be written as “1-3”. One can write any number of lines. We note that CafeMol reads through and treat these changes sequentially from the top line to the bottom. Thus, order of lines is of crucial importance. Always, the change specified in lower line has higher priority than the change in upper line.

The followings are only for very advanced use. When one tries to redefine the other parameters for hydrophobic interactions, for following parameters one can use a convenient way to assign the values.

```
coef_aa_hp(AA) = real ($\epsilon_{HP,AA}$ of hydrophobic interaction)
rnccoors AA_hp(AA) = real ($n_{AA}$ of hydrophobic interaction)
rncoormax AA_hp(AA) = real ($n_{\text{max},AA}$ of hydrophobic interaction)
cutoff_dmin AA_hp(AA1-AA2/AA3-AA4) = real ($r_{\text{min},AA1,AA2}$ of hydrophobic interaction)
cutoff_dmax AA_hp(AA1-AA2/AA3-AA4) = real ($r_{\text{max},AA1,AA2}$ of hydrophobic interaction)
```

where `AA` is the index of the amino acid as defined in the hydrophobic.para (See Chapter 4). For multiple amino acids, one can use the same style for expressing continuous numbers as described above, (AA$_1$-AA$_4$). Similarly, one can define the amino acid pairs by (AA$_1$-AA$_2$/AA$_3$-AA$_4$), in which all the pairs (AA$_i$, AA$_j$), where AA$_1$ $\leq$ AA$_i$ $\leq$ AA$_2$ and AA$_3$ $\leq$ AA$_j$ $\leq$ AA$_4$, are considered.

5.17 <<<<< md_information (required)

This input blocks defines the type of MD simulations one is performing. A number of parameters have to be specified, all of which use “name_of_parameter = value” style.

```
n_step_sim = integer (required)
```
which is 1 for normal runs. When you want to switch the potential \( m \) times in the run, i.e., switching Go model or switching multiple-Go model, the number of steps of the simulation is called \( m + 1 \). and thus “n_step_sim” is to be set as “\( n = m + 1 \).”

\[
n_{tstep}(integer) = long_{integer} \quad \text{(required)}
\]

which defines the number of MD steps in each step before a potential switch. You are required to write this line for “n_step_sim” times.

\[
tstep\_size = real \quad \text{(required)}
\]

which defines a time length in each MD step. A typical value would be 0.05, 0.1 or 0.2. For Langevin dynamics of proteins, 0.2 could be okay. For Newtonian dynamics as well as for systems that include nucleic acids, 0.1 or smaller value would be recommended.

\[
n_{step\_save} = integer \quad \text{(required)}
\]

which sets how often structural information is saved in movie files (movie, dcd, and vmdc formats).

\[
n_{step\_rst} = integer \quad \text{(optional)}
\]

which sets how often information is saved in restart files (rst).

\[
n_{step\_neighbor} = integer \quad \text{(required)}
\]

which defines how often the neighbor list is to be updated. A typical value would be 100.

\[
tempk = real \quad \text{(required)}
\]

which defines the temperature used in the simulation in (approximate) “Kelvin”.

\[
n_{seed} = integer \quad \text{(required)}
\]

which sets the seed of the random numbers (any 32-bit integer except for 0). The random numbers are used to make a random conformation (when required), to make a set of initial velocities, and to produce random forces in Langevin dynamics.

In Langevin dynamics, one needs to specify the friction coefficient \( \gamma \). In CafeMol one can set the value of \( \gamma \) in the block “<<< redefine para”.

There are several switches that control whether some treatment is done or not.
5.17.  "<<<<< MD_INFORMATION (REQUIRED)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value 1</th>
<th>Value 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_com_zeroing</td>
<td>0: no operation</td>
<td>1: fix the center of mass of the system at the origin</td>
</tr>
<tr>
<td></td>
<td>(recommended for Langevin and MPC dynamics)</td>
<td>(recommended for Newtonian dynamics)</td>
</tr>
<tr>
<td>i_no_trans_rot</td>
<td>0: no operation</td>
<td>1: prohibit overall translation and rotation of the entire molecular system</td>
</tr>
<tr>
<td></td>
<td>(recommended for Langevin and MPC dynamics)</td>
<td>(recommended for Newtonian dynamics)</td>
</tr>
</tbody>
</table>

Recommended values of these two switches are linked to the type of dynamics. In our earlier study of protein folding, we often observed that, when Newtonian dynamics was used, kinetic energy was gradually shifted from the internal degrees of freedom to the overall translational and/or rotational degrees of freedom, which resulted in extending the protein chain into nearly straight line (by this, with a small angular momentum, kinetic energy of rotation increases radically) and freezing the internal motion. This occurred because of some small non-zero angular momentum and thus it is highly recommended to prohibit overall translational and rotational motions when the Newtonian dynamics is used. Since the translational motion is inhibited, it is easier for the analysis to move the center of mass of the system to the origin of the space coordinate. On the other hand, Langevin dynamics does not have this problem and thus it is recommended not to use this kind of treatment.

The followings are optional parameters.

```
i_com_zeroing.ini = 0 : no operation (default) 1 : move the center of mass of initial structure to origin
```

When “i_com_zeroing.ini = 1”, the center of mass of initial structure moves to the origin before the simulation starts.

```
i_implig = 0 : no operation (default) 1 : use implicit ligand model with MD-MC scheme
```

Use of this option is suitable mostly for advanced users. All the parameters except those in hydrophobic.para and in electrostatic.para can be re-defined (changed).
CHAPTER 5. INPUT FILE: HOW TO MAKE

\begin{verbatim}
i_energy_para = 0 : no operation (default)
   1 : re-scale some energy parameters
       (requires the input block "<<<< energy_para"

This is used when you want to re-scale strength of some interactions for some portions in the simulated system. For example, Go-interactions between two units can be doubled relative to Go interaction for the intra-unit interaction by this option.

\begin{verbatim}
i_neigh_dist = 0 : no operation (default)
   1 : re-define radius threshold for making the neighbor list
       (requires the input block "<<<< neighbor_dist"

i_mass_fric = 0 : no operation (default)
   1 : re-define mass and/or friction coefficient of some portions
       (requires the input block "<<<< mass_fric"

i_del_int = 0 : no operation (default)
   1 : delete some interactions in a particular range
       (requires the input block "<<<< del_interaction"

This is particularly useful when some portions of a protein is disordered and thus is not seen in the PDB structure, but you still need to include these residues in the simulation. In this case, you may model these disordered loops by some software and use the modelled structure as the native. Yet, you do not want to rely heavily on these loop structure in the model. Then, it is recommended to delete Go-like interactions that are dependent on the modelled loops.

\begin{verbatim}
i_anchor = 0 : no operation (default)
   1 : anchor some mass-points by springs to some positions
       (requires the input block "<<<< anchor_para"

i_bridge = 0 : no operation (default)
   1 : to bridge two mass-points by springs
       (requires the input block "<<<< bridge_para"

i_pulling = 0 : no operation (default)
   1 : to pull mass-points by constant forces
       (requires the input block "<<<< pulling_para"

i_fix = 0 : no operation (default)
   1 : fix some mass-points to their initial positions
       (requires the input block "<<<< anchor_para"

i_in_box = 0 : no operation (default)
   1 : put the entire system into a box of rectangular solid
       (requires the input block "<<<< in_box"

When one wants to use the modified-multicanonical ensemble method (Gosavi et al., 2006), one needs to turn on the switch as

\begin{verbatim}
i_modified_muca = integer
   =0 for not using the modified multicanonical ensemble method (default)
   =1 for using the modified multicanonical ensemble method
\end{verbatim}
\end{verbatim}
5.18  <<<><< MPC_DYNAMICS (OPTIONAL)

This input block is required to set mpc related parameters only when the mpc dynamics is selected with "i_simulate_type=4" in "<<< job_cntl" block.

In MPC dynamics, the system is put in a virtual rectangular (ideally cubic) box with sides \((L_x, L_y, L_z)\). Ideally, this box size \((L_x, L_y, L_z)\) should be larger than the simulated system. The whole of simulation box for MPC dynamics is defined by the set of minimum and maximum coordinates for each sides \((x, y, z)\) [unit is \(\AA\)].

\[
\begin{align*}
\text{pbbox}_{\text{min}}_x &= \text{real} \quad \text{(required)}  \\
\text{pbbox}_{\text{max}}_x &= \text{real} \quad \text{(required, should be larger than pbbox}_{\text{min}}_x)  \\
\text{pbbox}_{\text{min}}_y &= \text{real} \quad \text{(required)}  \\
\text{pbbox}_{\text{max}}_y &= \text{real} \quad \text{(required, should be larger than pbbox}_{\text{min}}_y)  \\
\text{pbbox}_{\text{min}}_z &= \text{real} \quad \text{(required)}  \\
\text{pbbox}_{\text{max}}_z &= \text{real} \quad \text{(required, should be larger than pbbox}_{\text{min}}_z)
\end{align*}
\]

where \(\text{pbbox}_{\text{max}}_\alpha - \text{pbbox}_{\text{min}}_\alpha (\alpha = x, y, z)\) corresponds to the length of side \(L_\alpha\).

The grid number \((n_x, n_y, n_z)\) for each side is specified as

\[
\begin{align*}
\text{ngrid}_x &= \text{integer} \quad \text{(required)}  \\
\text{ngrid}_y &= \text{integer} \quad \text{(required)}  \\
\text{ngrid}_z &= \text{integer} \quad \text{(required)}
\end{align*}
\]

where \(L_\alpha/n_\alpha (\alpha = x, y, z)\) corresponds to the grid size of each side \(a_\alpha\).

The average number \(\gamma_s\) of MPC solvent particles is specified as

\[
\begin{align*}
\text{n_av_solvent} &= \text{integer} \quad \text{(required)}
\end{align*}
\]

where the total number \(N_s\) for MPC solvent particles is given by \(N_s = \gamma_s n_x n_y n_z\).

The step number for collision interval \(ci\) is specified as

\[
\begin{align*}
\text{n_step_collision} &= \text{integer} \quad \text{(required)}
\end{align*}
\]

where the time interval for collision phase \(\tau_s\) is given by \(\tau_s = ci \times h\) (\(h\) is time step for integration).

The interval step number for velocity rescaling to keep the temperature is defined as

\[
\begin{align*}
\text{nratio_vcorrect_step} &= \text{integer} \quad \text{(required)}
\end{align*}
\]

(Ideally \(\text{nratio_vcorrect_step}\) is the same as \(\text{n_step_collision}\).)

The mass \(m_s\) of MPC solvent particle and the rotation angle \(\alpha\) for the collision procedure are defined as

\[
\begin{align*}
\text{rmass_solvent} &= \text{real} \quad \text{(required)}  \\
\text{rotate_angle} &= \text{real} \quad \text{(required)}
\end{align*}
\]

where the unit of rotate_angle is degree not radian.
5.19 <<<<< implicit_ligand (optional)

This input block is required only when the implicit ligand model for MD-MC simulation is turned on with "i_implig=1" in "<<<< md_information". In this block, except for the residue list involved to ligand binding, 7 or 8 parameters need to be specified. The residue list involved to the ligand binding is specified in "<<<< binding_site" block.

At first, the system can have one or more ligands, and for each ligand we assume one ligand-binding-site. The total number of implicit ligands considered is specified as.

\[ \text{nsite_implig} = \text{integer} \quad \text{(required)} \]

For each implicit ligand binding site \( lbs \) (\( 1 \leq lbs \leq m \)), the initial state "initial_state_implig" which is either bound state (= 1) or unbound state (= 0) should be specified as.

\[
\begin{align*}
\text{initial_state_allimplig} & = \text{integer} \\
\text{initial_state_implig}(lbs_{ini} - lbs_{las}) & = \text{integer} \quad \text{(can be more than one line)}
\end{align*}
\]

where the first line can set the initial states for all the ligand binding sites at once. The second line sets (or modifies) the initial states for the specified ligand binding sites between \( lbs_{ini} \) and \( lbs_{las} \) (\( lbs_{ini} \leq lbs_{las} \)). With "nsite_implig = m", \( lbs_{ini} \) and \( lbs_{las} \) should be in the range \( 1 \leq lbs_{ini} \leq lbs_{las} \leq m \). If \( lbs_{ini} \) and \( lbs_{las} \) are equal to \( lbs \), the following description is also possible.

\[
\begin{align*}
\text{initial_state_implig}(lbs) & = \text{integer} \quad \text{(can be more than one line)}
\end{align*}
\]

Second, one needs to specify binding and unbinding rates and a related parameter. The ligand binding rate constant \( k_b \). (This takes into account and thus is proportional to the ligand concentration. Please check Chapter 3 for detail information.) is specified as

\[
\begin{align*}
\text{bind_rate_allimplig} & = \text{real} \\
\text{bind_rate_implig}(lbs_{ini} - lbs_{las}) & = \text{real} \quad \text{(can be more than one line)}
\end{align*}
\]

where the first line sets the binding rate constant \( k_b \) for all the ligand bindings. The second line sets (or modifies) \( k_b \) for the specified ligand binding sites between \( lbs_{ini} \) and \( lbs_{las} \) (\( lbs_{ini} \leq lbs_{las} \)). Technically the binding event is tried in a certain frequency. One needs to specifies the MD time step for this trial as

\[
\text{istep_implig} = \text{integer} \quad \text{(required)}
\]

which corresponds to \( \Delta t_b \). As far as this step is small enough, the result should not be affected by this, in statistical sense. On the other hand, for the ligand unbinding process, one specifies,

\[
\text{istep_un_implig} = \text{integer} \quad \text{(required)}
\]
that corresponds to the parameter $\Delta t_u$. (please see Chapter 3 for detail).

Then, one defines the interaction (energy) type for implicit ligand model.

```
itype_ene_implig = integer
    0 : LJ12-10 type interaction
    1 : Gaussian type interaction
```

When “itype_ene_implig= 1” (Gaussian type interaction) is used, for each ligand binding site ($lbs$) you need to specify not only “pre_implig” which is the control parameter ($c_{lig}$) that changes strength of ligand-mediated contact but also “gauss_d_implig” ($\sigma$) which defines the interaction range for ligand-mediated contact.

Finally, depending on the interaction energy form chosen, one needs to specify corresponding parameters. In case of itype_ene_implig = 1,

```
pre_allimplig = real
pre_implig(lbs_{ini} - lbs_{las}) = real
    (can be more than one line)
gauss_d_allimplig = real
    gauss_d_implig(lbs_{ini} - lbs_{las}) = real
    (can be more than one line)
```

The first and the third lines set the parameters $c_{lig}$ and $\sigma$ for the whole of ligand binding sites, respectively. The second and the fourth lines set (modify) the parameter $c_{lig}$ and $\sigma$ for the specified ligand binding sites between $lbs_{ini}$ and $lbs_{las}$ ($lbs_{ini} \leq lbs_{las}$).

In case of itype_ene_implig = 0,

```
pre_allimplig = real
pre_implig(lbs_{ini} - lbs_{las}) = real
```

The first line sets the parameter $c_{lig}$ for the whole of ligand binding sites. Second line sets (changes) the parameter $c_{lig}$ for the specified ligand binding sites between $lbs_{ini}$ and $lbs_{las}$ ($lbs_{ini} \leq lbs_{las}$). We note that one needs not to specify the parameter “gauss_d_implig” ($\sigma$).

### 5.20 <<<< binding_site (optional)

This input block is required for the implicit ligand. In particular, when “i_implig=1” in “<<<< md_information” block (implicit ligand model for MD-MC simulation), this block is required.

When “<<<< implicit_ligand” block (implicit ligand model for MD-MC simulation) is used, we have to specify the residues (not mass-points) and “unit & states” involved to ligand binding to fix the “ligand-mediated contact pairs” for calculating of ligand-binding energy. Since the ligand-mediated contact pairs are given based on the native structure of protein (see Chapter 3 for the detail definition of mediated contact), we have to specify not only the residue numbers list ($aa_1\ a2\ a_3\ldots$) but also “unit & states” ($u&\ s$) for each ligand binding site ($lbs$).

```
IMPLIGSITE lbs u&s_lbs u aa_{lbs,1} aa_{lbs,2} aa_{lbs,3} \ldots
```

(can be more than one line)
where \( lbs \) is the ligand binding site number, \( u \& s_{lbs} \) stands for “unit & states” for \( lbs \)-th ligand binding site, and \((aa_{lbs,1} \ aa_{lbs,2} \ aa_{lbs,3},\ldots)\) means amino acid residues list for \( lbs \)-th ligand binding site. The character ”\( u \)” must exist before the aa numbers indicating that the numbering is the intra-unit numbering. When the total number of implicit ligand binding site is to be set as “\( nsite_{implig} = m \)” in “\( \lllll \) implicit_ligand” block, at least the \( m \) lines for all of binding sites \((1 \leq lbs \leq m)\) should be specified.

We note again that in this format, the binding residues list \((aa_{lbs,1} \ aa_{lbs,2} \ aa_{lbs,3},\ldots)\) are not accumulated numbering but intra-unit numbering (please pay careful attention).

### 5.21 \( \lllll \) redefine_para (optional)

This input block is required only when “\( i_{\text{redef\_para}=1} \)” in “\( \lllll \) md\_information” block. Here, variety of input parameters can be modified from their default values. All lines are optional.

Some important examples are

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{cbd} )</td>
<td>real ( K_b ) in Clementi et al.’s Go model</td>
</tr>
<tr>
<td>( \text{cba} )</td>
<td>real ( K_\theta ) in Clementi et al.’s Go model</td>
</tr>
<tr>
<td>( \text{cdih}_1 )</td>
<td>real ( K_\phi^{(1)} ) in Clementi et al.’s Go model</td>
</tr>
<tr>
<td>( \text{cdih}_3 )</td>
<td>real ( K_\phi^{(3)} ) in Clementi et al.’s Go model</td>
</tr>
<tr>
<td>( \text{dfcontact} )</td>
<td>real (the threshold length in Å for defining native contacts)</td>
</tr>
<tr>
<td>( \text{catra_go1210} )</td>
<td>real ( \varepsilon_{go} ) in Clementi et al.’s Go model</td>
</tr>
<tr>
<td>( \text{coef_rep12} )</td>
<td>real ( \varepsilon_{ev} ) in Clementi et al.’s Go model</td>
</tr>
<tr>
<td>( \text{cdist_rep12} )</td>
<td>real ( d ) in Clementi et al.’s Go model</td>
</tr>
<tr>
<td>( \text{rneighbor_dist} )</td>
<td>real (the threshold length in Å for making a neighbor list)</td>
</tr>
<tr>
<td>( \text{fric_const} )</td>
<td>real (the friction coefficient ( \gamma ) in Langevin equation. The default is set in general_para as 0.25)</td>
</tr>
</tbody>
</table>

Note that all these parameters have their default values, and thus ones need to specify them only when ones want to change them from the defaults.

There are many many parameters that can be modified, here. Some parameters defined in “\( ./\text{para/} \)” directory can be re-defined in the current block (See Chapter 4). Note that this block and “\( \lllll \) native\_info\_simN” block are not in conflict. Thus, one can use both simultaneously. The resulting coefficients would be the product of the value indicated here and the factor indicated in the native\_info file.

### 5.22 \( \lllll \) energy\_para (optional)

This input block is required only when “\( i_{\text{energy\_para}=1} \)” in “\( \lllll \) md\_information”. All lines are optional.

This block re-scales some energy components, either for entire system, or for specified units.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{rlocal_all} )</td>
<td>real (rescale local interactions in Go model)</td>
</tr>
<tr>
<td>( \text{rlocal_unit}(unit_{ini-1} - unit_{las-1}/unit_{ini-2} - unit_{las-2}) )</td>
<td>real (rescale local interactions in Go model for specified units)</td>
</tr>
<tr>
<td>( \text{go_all} )</td>
<td>real (rescale Go contact energy in Go model)</td>
</tr>
<tr>
<td>( \text{go_unit}(unit_{ini-1} - unit_{las-1}/unit_{ini-2} - unit_{las-2}) )</td>
<td>real (rescale Go contact energy in Go model for specified units)</td>
</tr>
</tbody>
</table>
Note that these parameters indicate multiplying factors. Thus the value of unity means “no change”. “rlocal,” re-scales local interactions in Clementi’s Go model, while “go,” rescales non-local Go contact energies. For example, “go_unit(1/2)=1.2” means that Go interactions between the unit 1 and the unit 2 is scaled by 1.2 from the original value indicated by “cgo1210”, which is defined either in Parameter directory or in “<<< redefined_para” block.

Note that the re-scaling in this input block is in CONFLICT with the “<<< native_info_simN”(N=1, 2,...) input. When one uses the native_info style, one cannot re-scale the parameter by this block.

5.23 <<< neighbor_dist (optional)

This input block is required only when “i_neigh_dist=1” in “<<< md_information” block. This re-defined parameters used for the neighbor list either for entire system or for parts of the system.

| rdist_all         | = real (optional) |
| rdist_unit(unit_i-1-unit_las-1/unit_i-2-unit_las-2) | = real (optional) |

The first line redefines the threshold length in Å for the entire system. The second line, on the other hand, redefines the threshold specifically for the interactions between unit\_i-1-unit\_las-1 and unit\_i-2-unit\_las-2.

5.24 <<< mass_friction (optional) |

This input block is required only when “i_mass_fric=1” in the “<<< md_information” block. This input block is to modify the mass and/or friction of either all, by chain, or by particles.

| rmass_all         | = real (optional) |
| rmass_unit(unit_i-unit_las) | = real (optional) |
| rmass_mp(mp_i-mp_las) | = real (optional) |
| fric_all         | = real (optional) |
| fric_unit(unit_i-unit_las) | = real (optional) |
| fric_mp(mp_i-mp_las) | = real (optional) |

The first and the fourth lines change the mass and the friction coefficient for the entire system, respectively. The second and the fifth lines change the mass and the friction coefficient for the specified units in the system. The third and the sixth lines change the mass and the friction coefficient for the specified mass-points (particles) in the system. The mass-points should be specified by their serial number.

5.25 <<< del_interaction (optional)

This input block is required only when “i_del_int=1” in “<<< md_information” block. In this case, Go-like interactions can be deleted for certain segments in the simulated molecules. This option is particularly designed for the protein, of which reference crystal structures have missing residues in their flexible loops and, still, this flexible loops have some crucial roles. For such cases, one wants to include these loops explicitly in the simulation, but one does not have reliable and unique native structural information for these loops. Then, one of the best ways is to remove Go-like interaction related to the loops.

For removing the local Go interaction (bond-angle and dihedral-angle terms. Notably, bond-length interaction is not removed) in Go model and AICG model one can use,
where \( aa_{ini} \) and \( aa_{las} \) are residue numbers of the initial and final amino acids (for RNA, they are serial numbers of mass-points), between which local interactions are removed. Here, when the system contains more than one chains, \( aa \) has to be specified as all-in-one serial number. Local potential terms that use coordinate information of at least one residue in the range are removed. (For old users, DEL\_LGO was originally termed DEL\_BA\_DIH. The old name is still allowed, but not recommended). In the AICG2 model, even when some local Go potentials (i.e., 1-3 and 1-4 Gaussians) are deleted, flexible local potentials are remained.

For removing the non-local Go contact term in simple Go model as well as AICG models, we use

\[
\text{DEL\_GO}(aa_{ini,1}-aa_{las,1} / aa_{ini,2}-aa_{las,2})
\]

by which Go-like native contact interactions are removed for the contacts between any residue in the range \( aa_{ini,1}-aa_{las,1} \) and any residues in the range \( aa_{ini,2}-aa_{las,2} \). Once the non-local Go interaction is removed, automatically the standard excluded volume interaction is added.

### 5.26 <<<< anchor\_para (optional)

This input block is required only when “i\_anchor=1” in “<<<< md\_information” block. By anchoring, some mass-points are constrained to some positions by harmonic springs. For each mass-point to be constrained, one writes a line of

\[
\text{ANCH} \ i \ k \ i_0 \ x_0 \ y_0 \ z_0
\]

where \( i \) is integer and the following 5 parameters are real, and by this line, a spring of \( V_{anch} \) is added to the potential energy.

\[
V_{anch} = \begin{cases} 
  k_i (r_{i0} - l_0)^2 & (r_{i0} \geq l_0) \\
  0 & (r_{i0} < l_0)
\end{cases}
\]  

where

\[
r_{i0} = [(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2]^{1/2}
\]

### 5.27 <<<< bridge\_para (optional)

This input block is required only when “i\_bridge=1” in “<<<< md\_information” block. This is to bind two mass-points by a harmonic spring. For each bridge, one needs to write a line of

\[
\text{BRIDGE} \ i \ j \ k_{ij} \ l_0
\]

where \( i \) and \( j \) are all-in-one-style serial number of mass-points (integers) and the rests specifies the spring (real), and by this line, a spring of \( k_{ij} (r_{ij} - l_0)^2 \) is added to the potential energy if \( r_{ij} \geq l_0 \).
5.28  <<<<< pulling_para (optional)

This input block is required only when "i,pulling=1" in "<<<< md_information" block. By this some part of the system is pulled by force.

When one wants to pull a mass-point by a constant force, one writes

\[
\text{PULL CF } i \ f_x \ f_y \ f_z
\]

(can be more than one line)

where \(i\) is all-in-one-style serial number of mass-points (integer) and the following 3 parameters are real. By this a force of \(\vec{f} = (f_x, f_y, f_z)\) is applied to the \(i\)-th mass-point.

When one wants to pull a mass point with constant velocity, one writes

\[
\text{PULL CV } i \ k \ d \ v_x \ v_y \ v_z \ x \ y \ z
\]

(can be more than one line)

where \(i\) is all-in-one-style serial number of mass-points (integer) and the following 7 parameters are real. By this, an energy term, \(k_d^2(r_i - r_g)^2\) is added, where the position of a ghost particle \(r_g\) is defined by \(r_g = (x + v_xt, y + v_yt, z + v_zt)\).

5.29  <<<<< fix_para (optional)

This input block is required only when "i,fix=1" in "<<<< md_information" block. By this, we can fix some mass-points or some units to their initial positions. It allows two styles,

\[
\begin{align*}
\text{FIX UNIT} & (\text{unit}_{ini} - \text{unit}_{las}) \\
\text{FIX MP} & (\text{mp}_{ini} - \text{mp}_{las})
\end{align*}
\]

where the first line fixes a range of units between \(\text{unit}_{ini}\) and \(\text{unit}_{las}\), whereas the second line fixes a range of mass-points between \(\text{mp}_{ini}\) and \(\text{mp}_{las}\).

5.30  <<<<< in_box (optional)

This input block is required only when "i,in_box=1" in "<<<< md_information" block. In this case, the simulated system is put in a rectangular box that is centered at the Origin (Note that the boxing energy here is different from that in (Takagi et al., 2003). In this input block, the box size and sharpness of the wall is defined.

\[
\begin{align*}
x_{box} & = \text{real} \quad \text{(required)} \\
y_{box} & = \text{real} \quad \text{(required)} \\
z_{box} & = \text{real} \quad \text{(required)} \\
\text{boxsigma} & = \text{real} \quad \text{(required)}
\end{align*}
\]
where the first three lines define the rectangular box size in Å. The box is located in the range, \(-xbox/2 \leq x \leq xbox/2, -ybox/2 \leq y \leq ybox/2, \) and \(-zbox/2 \leq z \leq zbox/2.\) “boxsigma” represents the sharpness of the wall. The interaction between any particle with the wall is defined as

\[
V_{\text{box}} = \begin{cases} 
0 & (d > 3\sigma) \\
K_{\text{box}} (\frac{\sigma}{d})^{12} & (0.5\sigma < d < 3\sigma) \\
K_{\text{box}} (\frac{\sigma}{d})^{12} (1 + 12 \frac{0.5\sigma - d}{0.5\sigma}) & (d < 0.5\sigma) 
\end{cases}
\] (5.3)

where \(d\) is the distance between the particle and the wall, \(\sigma\) represents the sharpness of the wall, “boxsigma”, \(K_{\text{box}}\) is a large value (=10).

### 5.31 <<<<< searching tf (optional)

This input block is required only when “i_run_mode=4” in “<<< job_cntl” block. This is used to automatically search the folding transition temperature \(T_F\).

| tempk_upper   | = real | (required)                  |
| tempk_lower   | = real | (required, should be smaller than tempk_upper) |

These two temperatures provide the upper and lower bounds of the \(T_F\).

### 5.32 <<<<< annealing (optional)

This input block is required only when “i_run_mode=3” in “<<< job_cntl” block. This is to specify the parameters in the simulated annealing run. Parameters to be specified are the following three.

| tempk_init    | = real | (required)                  |
| tempk_last    | = real | (required, should be smaller than tempk_init) |
| n_time_change | = integer | (required)                  |

By this, the simulated run starts with the temperature =“tempk_init” (in Kelvin) and ends with the temperature =“tempk_last”. During the MD steps, the temperature is linearly decreased “n_time_change” times. The entire MD steps are evenly divided into “n_time_change+1” time windows, and the temperature for the first window is tempk_init and the temperature for the last window is tempk_last.

### 5.33 <<<<< replica (optional)

This input block is required only when “i_run_mode = 6 (REMD)” in “<<< job_cntl” block.

The following lines are required to specify the number of replicas.

| n_replica_temp | = integer | (at least one line is required) |
| n_replica_ion  | = integer |                                |

This field is important because it is the decision of which physical quantity is treated as replica-specifying variable. In the current version of CafeMol, two types of REMD variant, i.e., the temperature-REMD and the ionic-strength HREMD are implemented. One can perform either one-dimensional REMD using one of these two variants or two-dimensional REMD using both of them.
If one wants to perform one-dimensional REMD simulation, only one of the two lines should be used, and thus the other should be commented out. Meanwhile, if one wants to perform two-dimensional REMD which uses both temperature and ionic-strength as replica-specifying variables, both lines should be used.

The next line sets the frequency of exchanging trial event.

\[ \text{n\_step\_exchange} = \text{integer} \]  
(required)

Although some groups have suggested that one had better attempt exchanges as often as possible (Rosta and Hummer, 2009; Sindhikara et al., 2010), it is needed to compare the advantage of sampling efficiency with computational cost because more exchange events require more energy calculations which are heavy in some cases. We often use a value on the order of 10 - 100 (steps) just empirically.

The following line sets how often replica information is saved in rep file.

\[ \text{n\_step\_save\_rep} = \text{integer} \]  
(required)

For the convenience of data analysis after simulation, it is strongly recommended to set this value the same as “\text{n\_step\_save}” in “<<<< md\_information” block.

The following lines are related to the Optimized-Feedback REMD (FO-REMD). This is only preliminary, and not well documented.

\[ \text{i\_opt\_temp} = \text{0} : \text{not FO-REMD (default)} \]
\[ \text{= 1 : FO-REMD} \]  
(optional)

which specify whether FO-REMD is performed or not.

\[ \text{n\_step\_opt\_temp} = \text{integer} \]  
(optional)

which sets the number of initial interval steps for feedback iteration. This line is required only when “\text{i\_opt\_temp} = 1”.

The following line specify whether exchanges are performed or not.

\[ \text{i\_exchange} = \text{0} : \text{no exchange} \]
\[ \text{= 1 : exchange (default)} \]  
(optional)

This flag is originally prepared for debugging. When the value is 0, replica variables are never exchanged each other. It means that each replica is completely independent and performed as just normal (i.e. not REMD) simulation using their own replica variable at constant during simulation. For example, if one want to perform many canonical simulations at various temperature, one can do all of those simulations simultaneously by using this flag.

The following line is related to allocation of computational resources.
npar_rep = integer  (optional, only for advanced users)

Developers strongly recommend to just ignore this line and to keep it commented-out because CafeMol program can automatically determine the most efficient parallelization setting with given resources. Only advanced users who read the source code would use this line.

5.34  <<<< replica XXXX (optional)

“XXXX” should be replaced to “temp” or “ion” corresponding with temperature-REMD or ionic-strength HREMD, respectively.

This input block is required to define replica variables only when “i_run_mode = 6” in “<<<< jobcntl” block and “n_replica_temp (and/or n_replica_ion) > 1 in “<<<< replica” block.

There are three ways to generate a distribution of replica variables. The following line specifies how to do it.

```
i_style = 1 : Linear
10 : Exponential
100 : Explicitly define
```

In either the “Linear” or “Exponential” case, the following lines are required.

```
** Linear or Exponential
value_lowest = real
value_highest = real
```

which sets the lowest and highest value.

In the “Linear” case, the distribution is generated linearly between the two values.

\[ x_m = lowest + \frac{(highest - lowest)}{(M - 1)} \cdot (m - 1) \]  \hspace{1cm} (5.4)

In the “Exponential” case, it is generated by exponential interpolation.

\[ x_m = lowest \cdot \left( \frac{highest}{lowest} \right)^{\frac{m-1}{M-1}} \]  \hspace{1cm} (5.5)

Finally, in the “Explicitly define” case, all of replica variables should be set explicitly as following.

```
** Explicitly define
REPLICA(1) = real
REPLICA(2) = real
REPLICA(3) = real
...:
REPLICA(M) = real
```
5.35  \hspace{1cm} \texttt{modified\_muca} (optional)

When the modified multicanonical ensemble method is used with \texttt{i\_modified\_muca=1} in the \texttt{md\_information} block, this input block is required. This block sets the three parameters for the modified multicanonical ensemble method.

\begin{verbatim}
em_depth = real
em_mid = real
em_sigma = real
\end{verbatim}

where \texttt{em\_depth} is \(E_{\text{depth}}\), \texttt{em\_mid} is \(E_{\text{mid}}\), and \texttt{em\_sigma} is \(\sigma\) in the expression of the modified multicanonical ensemble method eq.(3.63). See the Method chapter for the expression.
Chapter 6

Examples: Input and output

Here, several examples of CafeMol runs are described. All the corresponding input and output files are stored in ./example/ directory. Comments and explanations for many input format are also found in cafemol_all.inp.

6.1 Protein native dynamics and (un)folding with Go model at constant $T$

The first example is constant-temperature MD of a protein, SH3 domain, with the Go model. The example file is, as it is, to perform constant temperature (300K) Langevin MD of SH3 domain, a small $\beta$-sheet protein domain with the friction coefficient $\gamma = 0.25$ starting from the native structure with pdb id=1srl.

6.1.1 Input & Output

The prepared input file is ./example/sh3/sh3.inp, and PDB file for initial structure is ./pdb/1SRL.pdb. Just by changing the initial state to “i_initial_state=1”, one can simulate folding from a random conformation. Instead, if one replaces the temperature with “tempk=500”, say, one would observe quick unfolding. The output files are given in the same directory.

6.2 Protein native dynamics and (un)folding with AICG model at constant $T$

This example is constant-temperature MD of the protein G using AICG2 energy function for both the local and nonlocal interactions. The Langevin MD with temperature of 300K was used. The parameters are prepared automatically by Cafemol with the option i_aicg = 1. One can also find the same type of run with the AICG1 model.

6.2.1 Input & Output

The prepared input file is ./example/aicg2/aicg2.inp, and PDB file for initial structure is ./pdb/2IGD.pdb. The output files are given in 2IGD.*, where * corresponds to “data”, “ninfo”, and “ts”.

One can conduct AICG1 simulations using the input file ./example/aicg/aicg.inp. In the case of AICG1 simulation, one needs to also provide the dssp file (2igd.dssp) containing the information of secondary structure in the directory ./aicg.
6.3 Automatic $T_F$ search

This example is to look for the folding transition temperature, $T_F$ at which the simulated protein reside in the native state for about half of the time and is unfolded for the rest of time. The target protein is still SH3 domain.

6.3.1 Input & Output

The prepared input file is ./example/sh3_stf/sh3_stf.inp, and PDB file for initial structure is ./pdb/1SRL.pdb.

The output files are given in ./example/sh3_stf/sh3_stf.*, where * corresponds to “data”, “ninfo”, and “ts”.

6.4 Conformational transition simulation with multiple basin model at constant temperature

This example conducts reversible closing and opening conformational changes of glutamine-binding protein with the multiple-basin potential.

As reference structures for multiple-basin model of glutamine-binding protein, we applied the open structure (pdb id=1GGG.PDB) and the closed structure (pdb id=1WDN.PDB) that are obtained by X-ray analysis.

6.4.1 Input & Output

The prepared input file is ./example/gbp_mgo1/gbp_mgo1.inp, and PDB file for initial structure is ./pdb/1WDN_2b.pdb.

The output files are given in ./example/gbp_mgo1.*, where * corresponds to “data”, “ninfo”, and “ts”.

6.5 Conformational transition simulation with multiple basin model and implicit-ligand (MD-MC scheme)

This is an example for a simulation in which the implicit-ligand model with MD-MC is applied (Okazaki and Takada, 2008). This simulates the protein conformational change which is coupled with (implicit) ligand binding. The protein used is the same as the above example, and is modeled by the multiple-basin potential.

As the initial state for implicit ligand, we set the bound state. Before the simulation, the residues involved implicit ligand binding was obtained by LIGPLOT (Wallace, Laskowski, and Thornton, 1995).

During about 10 million time steps, we can observe [several times] reversible conformational changes (a kind of equilibrium transitions) that are stochastically coupled with the ligand binding/unbinding processes. Note that the detail (concrete) parameters set for simulation is similar to, but not identical to, that in (Okazaki and Takada, 2008).

6.5.1 Input & Output

The prepared input file is ./example/cafemol_implig/cafemol_implig.inp, and PDB file for initial structure is ./pdb/1WDN_2b.pdb (closed structure). PDB files for reference structures are ./pdb/1GGG_2.pdb and ./pdb/1WDN_2.pdb corresponding to open and closed states, respectively. The initial state for implicit ligand is bound.

The output files are given in ./example/cafemol_implig/cafemol_implig.*, where * corresponds to “data”, “ninfo”, and “ts”.
6.6 Cyclic conformational change simulation by switching Go model at constant temperature

This is a simple example that employs “switching-Go” model. For the target protein of glutamine-binding protein, one starts the simulation using the Go model with the reference of the open conformation. At a certain time of simulation, one switches the reference structure into the closed one, which promptly induces large-scale conformation change of the protein.

Here, for switching, one uses the “native_info” file as one of the input data. For this purpose, “all-in-one-file” style is more convenient and is used in the example.

6.6.1 Input & Output

The prepared input file is ./example/cafemol_sw1chain/cafemol_sw1chain.inp, and PDB file for initial structure is ./pdb/1GGG_2.pdb. As aforementioned, while PDB file is not used for reference structure, ./ninfo/gbp1.ninfo and ./ninfo/gbp2.ninfo are used.

The output files are given in cafemol_sw1chain.*, where * corresponds to “data”, “ninfo”, and “ts”.

6.7 Rotary motion of F1-ATPase by switching Go model at constant temperature

This is an example for simulation of rotary motion of F1-ATPase (We note that the simulation scheme here is similar to, but not identical to, that in (Koga and Takada, 2006)). It uses a minimal catalytic set, α3β3γ of which X-ray structure of Walker (Abrahams et al., 1994)(pdb id=1bmf.pdf) is used as the reference. In the crystal structure, chain pairs CD, AE, and BF are three pairs of αβ-subunits, in which the nucleotide-states are different each other; β-subunit of chain D binds ADP, β-subunit of chain E is nucleotide-free (called empty), and β-subunit of chain F binds ATP-analog. Starting from this complex structure, one switches the reference structures of the three αβ pairs so that it drives chain CD into nucleotide-free state, chain AE into ATP-bound state, and chain BF into ADP-bound state. After switching the potential, conformations of the α3β3 change, which is followed by the rotation of the central stalk γ that corresponds to chain G.

For this case, reference structures of the αβ-pair is permuted. To realize it, the so-called “one-by-one-file” style “native_info” files are utilized. We note that, with the current setting, the rotation of the central stalk γ would occur with probability of about 0.5.

6.7.1 Input & Output

The prepared input file is ./example/f1atp_sw1onebyone/f1atp_sw1onebyone.inp, and PDB files for initial structure are ./pdb/gamma.pdb and ./pdb/#_*_.pdb where # corresponds to “alpha” and “beta”, and * corresponds to “E” and “TP”, and “DP”. The “native_info” files are given in ./ninfo/f1atp_*_.ninfo, where * corresponds to “alpha_E”, “alpha_TP”, “alpha_DP”, “beta_E”, “beta_TP”, “beta_DP”, “gamma”, “alphE_betaE”, “alphE_betaTP”, “alphE_betaDP”, “alphTP_betaE”, “alphTP_betaTP”, “alphTP_betaDP”, and “alphDP_betaE”.

The output files are given in ./example/f1atp_sw1onebyone/f1atp_sw1onebyone.*, where * corresponds to “data”, “ninfo”, and “ts”.

6.8 Conformational change and an explicit ligand(drug) dissociation in a multidrug transporter by multiple-basin model at constant temperature

This is an example to simulate the conformational change of a transporter (particularly the multidrug resistant transporter AcrB) (one subunit and the porter domain only) by multiple-basin model (3-basin case)
with explicit ligand (drug), and the subsequent drug dissociation from the binding pocket. The simulation is based on Yao et al 2010(Yao et al., 2010), but not identical to this reference. The drug molecule is modeled as a rigid linear chain (by setting the “ligand” class for the corresponding unit in the input file), and the hydrophobic energy term is included for the interactions between drug and protein. The fiducial structures and initial coordinates for both protein and drug are generated from the X-ray structure of Murakami et al (Murakami et al., 2006)(pdb id=2DRD). The simulation contains two stages: In stage I the stabilities of the “binding”, “extrusion”, and “access” states of the subunit are tuned to be equal, and therefore the extra binding energy between drug and protein stabilizes the system in the “binding” state. Then, in stage II, the energy of the “extrusion” state is artificially changed by -10.0, resulting in a quick conformational change of protein from “binding” to “extrusion”. In “extrusion” state, the drug is weakly bound and dissociates soon later. The simulation is performed by using Newtonian dynamics and the Berendsen’s thermostat. To prevent the overflow of coordinates, the distance between two atoms, belonging to protein and drug respectively, is constrained within 900 Å. We also note that the simulation is stochastic and the successful drug dissociation would occur with some probability less than unity.

6.8.1 Input & Output

The prepared input file is ./example/cafemol_ligand_hp/cafemol_ligand_hp.inp. The PDB files for initial structure are ./pdb/2drd_portera.pdb, ./pdb/2drd_porterb.pdb, ./pdb/2drd_porterc.pdb, and ./pdb/2drd_ligand.pdb. The first three files containing the coordinates of the porter domains of AcrB are generated respectively from the chain A (“binding” state), B (“extrusion” state), and C (“access” state) of the 2DRD structure. The fourth PDB file is constructed by selecting six representing atoms from the ligand model in the 2DRD structure. The “native_info” file is given in ./ninfo/acrb.ninfo. Here, we use the “all-in-one” style “native_info” file.

The output files are given in ./example/cafemol_ligand_hp/cafemol_ligand_hp.*, where * corresponds to “data”, “ninfo”, and “ts”.

6.9 Protein-protein docking with electrostatic interactions

This is an example to use the electrostatic interactions between two proteins; calmodulin and its binding peptide. Each of two proteins are modeled by the standard Go model, while the interaction between the two proteins are purely electrostatic. In the initial state, two proteins are separated by about 40 Å or so, but the electrostatic interactions attract them and make a complex. Yet, because of the lack of specificity, they do not make specific complex structure.

6.9.1 Input & Output

The prepared input file is ./example/calmodulin/calmodulin.inp. The result files are calmodulin.data, calmodulin.ts, and calmodulin.ninfo.

6.10 Temperature replica exchange molecular dynamics simulations

This is a simple example to run the replica exchange MD (REMD). Here, each replica differs in temperature, and thus it is the so-called temperature REMD. The example is to conduct REMD for protein G with the Clementi et al’s Go model for 8 replicas distributing exponentially in the range between 300K and 380K.

6.10.1 Input & Output

The prepared input file is ./example/cafemol_go_replica/cafemol_go_replica.inp.
6.11 tRNA

This is an example to run a 300K constant-temperature simulation of tRNA.

6.11.1 Input & Output

The prepared input file is ./example/trna/trna.inp and required PDB file is ./pdb/1ehz_min.pdb.

6.12 DNA

This example is a constant temperature simulation of 130bp DNA duplex.

6.12.1 Input & Output

The prepared input file is ./example/dna130/dna130.inp and any PDB file is not required. In this case, sequence of DNA duplex is written in the input file, and automatically make B-type DNA structure as for initial structure.

6.13 Using coarse-grained representation as an initial structure

One can use coarse-grained molecule as an initial structure instead of all-atom coordinates. In this case, it is necessary to use native information (.ninfo) at the same time because it should be constructed by all-atom coordinates.

6.13.1 Input & Output

The prepared input file is ./example/trna_readcg/trna_readcg.inp and required PDB file is ./pdb/1ehz_min_cg.pdb which is already coarse-grained. Additionally ./ninfo/trna.ninfo is needed to specify native interactions.

6.14 p53 searching its specific binding site on DNA simulated using flexible local potential

p53 is a famous transcription factor which have tumor suppressor activity. This is an example of the simulation of proteins which have large intrinsically disordered regions e.g. p53. These regions are modeled by the flexible local potential. This example also contains DNA. The interaction between the protein and the DNA is purely electrostatic interaction. Nonetheless, p53 can spontaneously bind to DNA and diffuse along it after long simulation.

6.14.1 Input & Output

The prepared input file is ./example/p53_DNA_flexible_local/p53_DNA_flexible_local.inp and required PDB file is ./pdb/p53.pdb and ./pdb/DNA200_for_p53.pdb. Additionally ./ninfo/p53-init.ninfo is needed to specify native interactions.
6.15 Reversible dissociation and association of a protein heterodimer with MPC dynamics

This is an example to simulate the reversible (intermittent) dissociation and association in a dimeric protein (Barnase-Barstar complex) by MPC dynamics. The reference structures and initial coordinates for the dimeric protein are the X-ray structure of the complex; chain A and B of pdb id=2ZA4. The default parameter sets are applied to the intra-chain interaction, while the weaker Go-interaction (0.6 times as strong as the default) is applied to the inter-chain interaction to simulate prompt dissociation and association at the room temperature (300 K).

To reproduce the suitable diffusion coefficient of protein in water environment and the appropriate stability of protein (which is measured by the folding temperature), we set MPC related parameter as follows: step size for time integration $h = 0.01$, step number for collision interval $c_i = 20$, the grid size $(L_\alpha/n_\alpha = 4\AA$ $[\alpha = x, y, z])$, the mass of solvent particle $m_s = 1.0$, the average solvent particle number for each grid $\gamma_s = 8$, and rotation angle for collision procedure $\alpha = 90.0$. To enhance the association rate, we applied a bridge potential (distance restraint) between the representative mass point for each subunit (chain A and B). [MPC box-size $(L_x, L_y, L_z = 72\AA)$ is large enough to the size of dimeric protein (the distance along long axis of dimer $\sim 51\AA$ [in $\alpha$ based native structure]) and the restraint length $l_0 = 40\AA$ for bridge-function.]

In our example, the step number for time integral is set as $10^9$. (To observe the several dissociation and association events, the step number for time integral must be at least larger than $10^9$.)

6.15.1 Input & Output

The prepared input file is ./example/mpc/mpc.inp, and required PDB file is ./pdb/2ZA4_AB.pdb and ./pdb/2ZA4_ini.pdb. The result files are ./example/mpc/mpc.data, ./example/mpc/mpc.ts, and ./example/mpc/mpc.ninfo. (In mpc.ts, there are not whole of time series, but a part of results.)
Chapter 7

Utilities: How to use

7.1 Post-MD analysis

In this section various programs for analyzing simulated trajectory are described.

7.1.1 Calculation of RMSD

The time series of the Root-Mean-Square-Deviation (RMSD) of a trajectory, compared to a reference structure, can be calculated by the following program,

\[
./cafe\_calc\_rmsd \ [REFERENCE\_FILE] \ [TRAJECTORY\_FILE] \ [OUTPUT\_FILE] \ ([INITIAL\ NUMBER] \ [LAST\ NUMBER])
\]

The reference file and the trajectory file should be in PDB and DCD format, respectively. The output file, saved in DCD format, contains the coordinates of trajectory after rigid-body superimposition. Initial and last numbers restrict the range of particles involved in the calculation. They should be specified in the format “uX” or “mX”, where X is a number and “u” or “m” indicates that the number refers to “unit” or “mass point”, respectively.

7.1.2 Making movie

A high-quality movie for a CafeMol simulated trajectory can be made by using the molecular graphics visualization software PyMol and the video encoding program Mencoder. An example is provided to make the movie for the simulation result of the example 6.8. The input file is

\[./example/cafemol\_ligand\_hp/cafemol\_ligand\_hp.movie.\]

The script to run is

\[./utility/cafe\_movie\_making.sh,\]

which will call the PyMol script

\[./utility/cafe\_movie\_making\_gengraph.pml\]

to generate static graphics. To run, type:

\[./utility/cafe\_movie\_making.sh\]

The output file ./cafe_movie.avi can be opened by many popular DVD players. Note that both script files (.sh and .pml) may need to be edited for user’s specific trajectories.
7.1.3 Application of single histogram re-weighting

To be added.

7.1.4 Application of WHAM

To be added.

7.2 Data conversion

7.2.1 show_rst and a2rst

The restart file (.rst) of CafeMol is recorded in binary format. The utility, show_rst, converts a binary .rst file to a text format so that the contents can be recognizable to users.

\$show\_rst hoge.rst

This command shows data of the rst file to standard output, normally just to a screen. Therefore if an output file is desired, one can uses redirect command as below,

\$show\_rst hoge.rst > hoge.restart

Here .restart file is written in text (normally ASCII) format.

In an opposite manner, a2rst converts a text format restart informaiton to the original binary restart file format (.rst).

\$a2rst hoge.restart hoge.rst
Bibliography


