分子動力学シミュレーションプログラム

MARBLE

マニュアル

ver. 0.6.1.0

このマニュアルについて

このマニュアルは、分子動力学シミュレーションプログラム MARBLE を利用するためものですが、まだベータ版であり、チュートリアル中心の構成となっています。今後、さまざまな計算例をチュートリアルとして追加すると同時に、MARBLE および、前処理用プログラムである molx のコマンドの記述も追加してゆく予定です。不備な点もまだ多く見られると思いますので、ご指摘いただけると幸いです。

2012 年 9 月 29 日

MARBLE マニュアル編集委員会

マニュアル（目次）

1　イントロダクション

1.1　MARBLE について

MARBLE(**M**olecul**AR** simulation program for **B**iomo**LE**cules)は、蛋白質をはじめとする生体高分子のシミュレーションを目的として開発された分子シミュレーションプログラムです。

以下のような特徴があります。

シンプレクティック部分剛体時間積分法を採用し、精度の高い全エネルギーの保存を実現しています。

長距離相互作用の計算に標準的なアルゴリズムである PME(Particle Mesh Ewald)を装備しています。

並列化プログラミングである OpenMP に対応し、系の空間分割による並列化を行います。

1.2　ライセンス

MARBLE のライセンスは、GPL (GNU General Public License) に準拠しています。

1.3　引用

MARBLE を利用して得られた成果を公表する際には、以下の文献を引用してください。

Ikeguchi M (2004) Partial rigid-body dynamics in NPT, NPAT and NPγT ensembles for proteins and membranes. J Comput Chem 25(4):529-541.

2　インストール

以下に、具体例として、いくつかのマシンに関してのインストール方法の具体例および一般的なインストール方法を示します。（”$”はプロンプトを意味します。）

「京」コンピュータでのコンパイル方法。

$ tar xvfz marble-x.x.x.tar.gz
$ cd marble-x.x.x/src
$ ln –s Makefile.machine.K　Makefile.machine
(ここで、Makefile.machine.K は「京」コンピュータ用のファイルです。)
$ make
$ make install

ここまで実行すると、marble-x.x/bin/に、実行ファイルとして marble.x.x.x_K と molx.x.x.x_K が生成されます。
＊ここで生成されるファイルはすべて、計算ノードに投入するための実行ファイルです。

FX10 でのコンパイル方法

$ tar　xvfz　marble-x.x.x.tar.gz
$ cd marble-x.x.x/src
$ ln –s　Makefile.machine.FX10　Makefile.machine
(利用する環境によって、module コマンドで以下のように FFTW をロードする必要があります)
$module load fftw
$ make
$ make install

ここまで実行すると、marble-x.x/bin/に、実行ファイルとして marble.x.x.x_FX10 と molx.x.x.x_FX10 が生成されます。
＊ここで生成されるファイルはすべて、計算ノード用の実行ファイルです。

Cray XE6 へのインストール方法

$ tar xvfz **MARBLE**-x.x.x.tar.gz
$ cd **MARBLE**-x.x.x/src
$ cd src
$ ln -s Makefile.machine.cray　Makefile.machine
(ここで、Makefile.machine.cray は Cray XE6 用のファイルです。)
$ module load PrgEnv-cray
$ module load fftw

```
$ make
$ make install
```

ここまで実行すると、marble-x.x/bin/に、実行ファイルとして marble.x.x.x-cray と molx-x.x.x-cray が生成されます。
＊ここで生成される marble.x.x.x-cray は、計算ノード用の実行ファイルです。
＊ここで生成される molx.x.x.x-cray は、計算ノード用の実行ファイルです。

そのほかのコンピュータへのインストール方法

現在のところ、上述の 3 つの計算環境でのみ実行可能であることのチェックを行っておりません。しかし、**MARBLE** は C 言語、OpenMP, MPI および FFTW3 を用いて記述されており、多くの並列計算機の上で **MARBLE** を実行することは可能であると思います。もし、MARBLE を上述以外の環境にインストールする際には以下の手順で行ってください。

(1) FFTW3 をインストール。
まず、インストールを行う環境に、FFTW3 がインストールされているかチェックします。もしインストールされていれば、コンパイルの際の利用方法をマニュアル等で確認して、次の”2. Makefile.machine ファイルの修正”を行ってください。
インストールされていない場合、以下のサイトから FFTW 3.x をダウンロード・インストールします。

http://www.fftw.org/

(2) Makefile.machine ファイルの修正
次に、marble-x.x.x/src にある、Makefile.machine.x（x=intel, gnu）を Makefile.machine というファイル名でコピーし、このファイルを、インストールするシステム用に修正します。Makefile.machine の内容は、以下のとおりです。

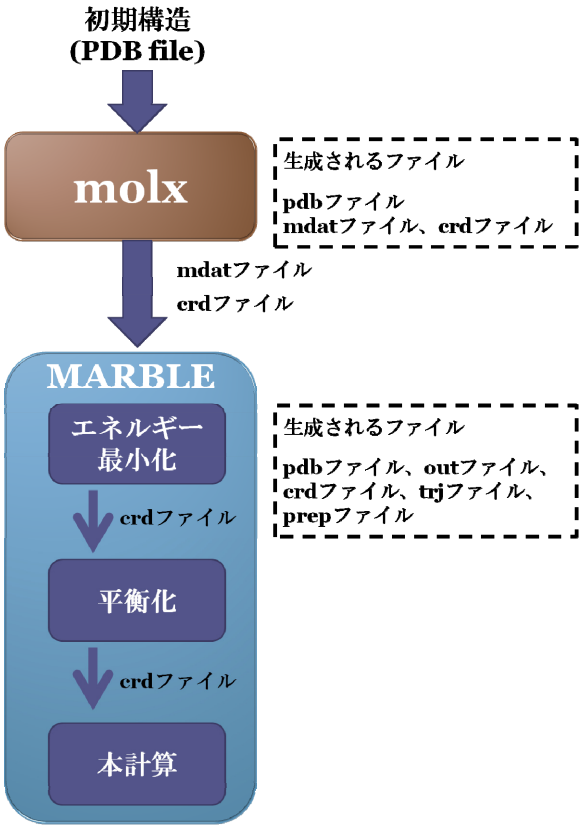
```
> more Makefile.machine
#
#   Makefile Setting for icc + openmpi
#


# for parallel programs
PCC             = mpicc                 # MPI プログラム用の C コンパイラー
PCOPTFLAG       = -std=gnu99 -O3 -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE   #最適化のオ
 プション
PCOMPFLAG       = -openmp -D_OPENMP     # OpenMP のオプション
PLD             = mpicc                 # MPI プログラム用のリンカー
PLIBFLAG        =                       # リンクするライブラリのフラグ  (e.g., -lm)
PARCH           = -intel                # コンパイルするシステムのサフィックス

# for serial programs
CC              = icc                   # シリアルプログラム用の C コンパイラー
COPTFLAG        = $(PCOPTFLAG)          # シリアルプログラム用の最適化オプション
LD              = icc                   # シリアルプログラム用のリンカー
LDFLAG          =                       # シリアルプログラム用のリンカーフラグ
LIBFLAG         =                       # リンクするライブラリのフラグ  (e.g., -lm)
LIBDIR          =                       # リンクするライブラリのディレクトリーのフラグ
ARCH            = $(PARCH)              # コンパイルするシステムのサフィックス

MARBLEHOME      = ../..
BINDIR          = $(MARBLEHOME)/bin     # 実行ファイルのインストールディレクトリ
DATDIR          = $(MARBLEHOME)/data    # データファイルのインストールディレクトリ

# for FFTW
FFTW_DIR        = /home/xxx/pub/fftw-3.3.2-install    # FFTW のインストール先
FFTW_INCLUDE    = $(FFTW_DIR)/include                 # FFTW のヘッダディレクトリ
FFTW_LIBDIR     = $(FFTW_DIR)/lib                     # FFTW のライブラリディレクトリ
FFTW_LIB        = $(FFTW_LIBDIR)/libfftw3.a           # FFTW ライブラリ
```

(3) make と make　install の実行

3　　チュートリアル

3.1　　計算の流れ

MARBLE を用いた分子シミュレーションは、以下のような流れで計算を行います。
　まず、計算を行う分子の構造情報を PDB format のファイル(Protein Data Bank 用いられる構造情報の書式のファイル。これ以降 pdb ファイルと呼ぶ）で用意します。



MARBLE での計算の流れ

3.1.1 molx の実行

使用方法：molx　インプットファイル
プログラム **molx** を用いて、分子シミュレーションを行う系を構築し、**MARBLE** での計算に必要な、構造情報および力場のパラメータ情報などを含んだファイルを生成します。

molx では、計算を行う分子の pdb ファイルをもとに、以下の操作を行います。
計算を行う分子に水素を付加、ジスルフィド結合などを定義。(モデルビルディング)
周期境界の箱を定義し、水分子やイオンなどを加えて分子シミュレーションを行う系を構築（システムビルディング）。

3.1.2 molx で生成されるファイル

molx を実行すると、構築された系に関する情報を含んだ以下のファイルを生成します。

pdb ファイル
構築された系全体の座標情報を含む pdb ファイルです。

crd ファイル
構築された系全体の座標情報を含むファイルです。**MARBLE** はこのファイルに含まれる座標情報を用いて計算を行います。

mdat ファイル
構築された系で分子動力学シミュレーションを行う際に用いる力場のパラメータ情報が含まれるファイルです。

3.1.3 MARBLE の実行

使用方法：MARBLE　インプットファイル　アウトプットファイル

molx で得られた、crd ファイルと mdat ファイルを用いて、**MARBLE** による分子シミュレーションを実行します。

MARBLE による計算は、おもに以下の 3 つのステップで計算を行います。

(1) エネルギー最小化
(2) 平衡化
(3) 本計算（Production run）

これらの計算は、前の計算で得られた crd ファイルと **molx** で得られた mdat ファイルを用いて行います。

3.1.4 MARBLE で生成されるファイル

MARBLE を実行することにより、以下のファイルが生成されます。

pdb ファイル
MARBLE 実行時の系の最終構造の座標情報です。

crd ファイル
MARBLE 実行時の系の最終構造の座標情報（ただし、分子動力学シミュレーション実行時には、座標情報に加えて最終構造の速度、シミュレーションで用いたアンサンブル、周期境界の箱の情報、温度・圧力コントロールに必要なパラメータなどが含まれます）が出力されます。**MARBLE** を用いてシミュレーションを引き続き行う際に、このファイルの情報を用います。

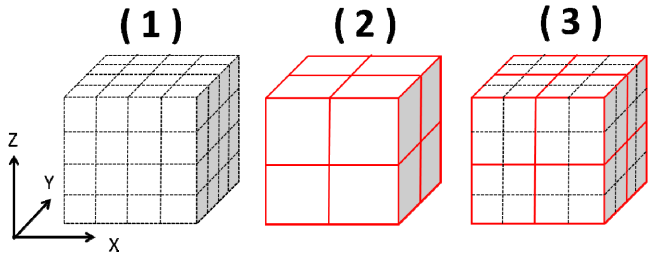
trj ファイル
分子動力学シミュレーションを行った際の系の経時変化情報が含まれています。情報として、系の座標情報・速度の情報・周期境界の箱の情報を含むことができます。trj ファイルに記録する情報内容および、情報を記録する時間間隔は、インプットファイルから指定することができます。

out ファイル
実行した計算内容とエネルギー、圧力および温度、計算の実行速度などの情報が含まれています。これらの情報を記録する間隔は、インプットファイルから指定することができます。

prop ファイル
計算途中のさまざまな変化量（一部 out ファイルの内容と重なるものもあります）を出力します。

3.1.5 MARBLE の並列化計算について

MARBLE は MPI 並列と openMP 並列を用いたハイブリッド並列による計算を前提に作られています。**MARBLE** でシミュレーション計算を実行する際には、以下のような並列処理を行っています（下図）。



MARBLE でのセル分割
およびプロセスの配置

- シミュレーションを行う系を X, Y, Z 方向に等間隔に分けることで系の空間を複数のセルに分割します（図(1)、点線で囲まれているキューブ）。
- 並列計算に用いるプロセスを X, Y, Z 方向に配置します（図(2)、赤線で囲まれているキューブ）。
- 各プロセスが、配置された場所に対応する、空間上に隣接する複数のセルの計算を担当するように分割します。

以上のような並列処理の分割を行い、通信を隣接セル間の情報のみに制限することで、シミュレーションの高速化を図っています。
従って、**MARBLE** で計算を行う際には、次の情報の指定が必要になります。

(a)系の XYZ 方向のセルの分割数
(b)利用するプロセスの数
(c)プロセスの XYZ 方向の配置の数

つまり、(a)により、上の図の(1)を、(b), (c)により上の図の(2)を決定します。

また、上述の(a)～(c)に加えて、

(d)PME(Particle Mesh Ewald)法による静電相互作用の計算の際に、空間を等間隔に分割したグリッドの XYZ 方向の数

が、必要です。

(a)～(d)の情報は、互いに特定のルールを満たすように決定する必要があります。

MARBLE で計算を行う際にこれらの条件を設定する方法は、以下の 2 つがあります。

(1)d_grid を設定する方法

まず、(b)のプロセスの数と PME のグリッドの定義に用いるグリッド間隔の値（d_grid）を指定することで、(a), (c)および(d)の情報を自動的に設定して計算を行なわせる方法です。この方法は、簡単に上述の(a)～(d)の情報を決定することができます。しかし、この方法を用いると、NPT アンサンブルのように系のボックスサイズが変化する場合、PME で用いるグリッドの数およびセルの分割数が変化する可能性があり、このことにより分子シミュレーションを異なるインプットファイルで、引き継いで計算させる場合などに、エネルギーが保存されなくなる恐れがあります。

(2)直接(a)～(d)の情報をインプットファイルに入力する方法

上述のように、d_grid を用いると、簡単に設定できる代わりに、ボックスサイズの変化に伴ってグリッド数が変化することがあるので、それを避けるために，直接(a)～(d)の情報をインプットファイルに入力する方法があります。

以上の並列計算の実行上の設定および、実行方法の例は、第 4 章に示してあります。

ただし、プロセス数の指定および、MPI 並列でのジョブの投入方法に関しては、用いる計算機のシステムにより異なりますので、それぞれの計算機のシステムのガイドなどを参考にしてください。

3.2　molx による系の構築

3.2.1　molx を実行する前に

MARBLE は、基本的に CHARMM 力場による分子シミュレーションを行うように、設計されています。したがって、**MARBLE** を用いる際には、以下のサイトから、必要な CHARMM 力場をダウンロードする必要があります。

http://mackerell.umaryland.edu/CHARMM_ff_params.html

molx は **MARBLE** を実行するために必要な 2 つのファイル（mdat ファイルと crd ファイル）を生成するために、系の座標の情報を必要とします。この座標の情報は、理想的には、水分子、イオンおよび水素原子を含む蛋白質分子のすべての原子の情報、さらに周期境界条件の箱の情報があることが望ましいです。（したがって、もし外部のプログラムを使ってこれらの情報をすべて含んだ pdb ファイルを作成してあるのであれば、**molx** はその情報から直ちに mdat ファイルと crd ファイルを生成することができます。）

しかし、実際に計算を始めるときに、計算したい蛋白質分子の構造も、水素原子まで含んで完全にそろっていることはほとんどありません。まして、計算をさせる系の水溶液の情報などまで持っていることは稀です。さらに、実際の蛋白質の構造では、ジスルフィド結合のような、特別な化学結合が形成されている場合があり、それらの情報を補ってやる必要があります。

molx は計算に必要なこれらの欠損した情報をある程度補ってモデリングを行う機能（モデルビルディング・システムビルディング）を持っています。

molx で行うことができるモデリングの機能は以下の通りです。

(1) 蛋白質分子の構造中で欠損している原子を補う（モデルビルディング）
molx は、すべての水素原子、および側鎖原子の欠損について、CHARMM 力場で用意されている各アミノ酸の構造テンプレートに基づいて、原子を捕います。

(2) ジスルフィド結合などの化学修飾を定義する（モデルビルディング）
蛋白質の種類によって、アミノ酸が様々な分子と結合している、あるいは、ジスルフィド結合のようにアミノ酸同士が結合しているような化学修飾を生じている場合があります。**molx** は、これらの結合について patch というコマンドを用いてこれらの化学修飾をセットアップすることが可能です。

(3) 蛋白質分子周辺に水分子およびイオンを発生させて水溶液系を構築する
（システムビルディング）

molx は周期境界条件の箱を定義して中心に蛋白質を置き、さらに水分子、イオンを箱中に発生させて蛋白質―水溶液系を生成します。

したがって、molx を実行する前に、蛋白質分子の構造について、おもに以下のことを確認することが必要となります。

(a) 蛋白質分子の原子欠損の有無

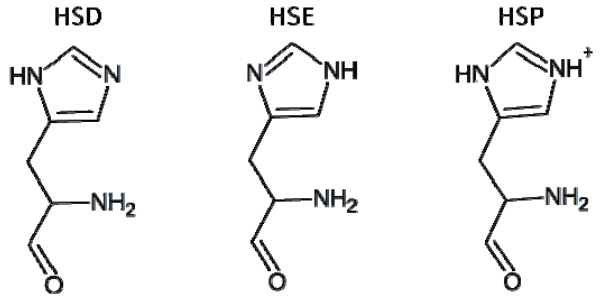
水素原子以外の重原子について、欠損の有無をチェックします。特に、主鎖も含めて原子の座標情報が存在しない領域が見られる場合、**molx** は原子を正しく補うことができない可能性があります。従って、そのような領域については、**molx** を実行する前に、モデリングを行う外部のプログラム（たとえば modeller 等）を用いて、欠損した領域を補っておく必要があります。

(b) マルチコンフォマーの有無

蛋白質の結晶構造の中には、マルチコンフォマーとして、側鎖のコンフォメーションを複数記述している場合があります。このような場合、**molx** はどの側鎖の情報を利用すればよいのか判断できません。そこで、どのコンフォメーションを採用するのかを決めて、pdb ファイルを編集しておく必要があります。

(c) アミノ酸のプロトン化状態の決定

荷電アミノ酸や極性アミノ酸（アスパラギン酸、グルタミン酸、リジン、アルギニン、ヒスチジン）では、アミノ酸側鎖のプロトン化状態が複数存在し、アミノ酸の存在する局所的な環境の影響で、その状態は変化します。従って、計算を行う前に、このようなアミノ酸の側鎖について、どのようなプロトン化状態を用いるのかを決めておく必要があります。とくに、ヒスチジンについては、側鎖の電荷の中性な 2 つの状態が存在し、周辺原子との水素結合の形成などで、違いが存在するので注意が必要です（下の図）。



ヒスチジンのプロトン化状態の例

(d) ジスルフィド結合の有無

molx は pdb ファイルからの座標情報のみでは、ジスルフィド結合の有無を決定できません。従って、pdb ファイルの SSBOND の情報などを参考にして、ジスルフィド結合の有無をチェックする必要があります。

3.2.2 molx 計算例ー１：リゾチーム

ここではリゾチームの結晶構造(PDB_ID:193L)について、**molx** を実行する場合の例を示します。

3.2.2.1 molx を実行する前に

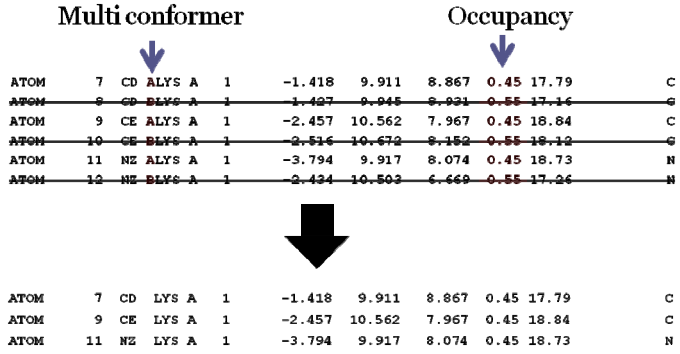
まず、この前の章の"**3.2.1 molx を実行する前に**"に示した、チェック項目(a)～(d)について確認します。

(a)蛋白質分子の原子欠損の有無

193L に重原子の欠損は存在しません。

(b)マルチコンフォマーの有無

193L の構造には LYS1, ASN59, SER86 および VAL109 について、それぞれ、２つのコンフォマーが存在します。ここでは、OCCUPANCY の値を参考にして、それぞれ、A のコンフォマーを選択して、PDB ファイルを修正します。下に、LYS1 を修正する際の例を示します。



マルチコンフォマーの修正

(c) アミノ酸のプロトン化状態の決定

存在する荷電アミノ酸はすべて電荷のある状態に定義します。さらに、ヒスチジンについては、"**molx** を実行する前に"に示したように、3 つのプロトン化状態があり、そのうち HSD と HSE は側鎖の電荷が中性です。193L の構造では HIS51 が存在しますが、ここでは、HSE とします。

(d) ジスルフィド結合の有無

193L の場合は、以下に示すように PDB ファイルのヘッダの SSBOND の記述からジスルフィド結合が 4 か所に存在することが分かります。

3.2.2.2 molx の実行

以上のことを考慮して、**molx** を実行するためのインプットファイルを作成します。
以下に、リゾチームの X 線構造の PDB フォーマットファイル(193L.pdb)を用いた水中での系を構築する際の例 を示します。
（以下の例の中で、">"で示されているのは、コマンドプロンプトです。）

```
> more molx2.in
#力場#
charmm_top_file ../../toppar/top_all27_prot_na.rtf
charmm_par_file ../../toppar/par_all27_prot_na.prm

#入力#
input_pdb_file     ../../pdbfile/193L.pdb

#出力#
output_mdat_file 193L_w.mdat
output_crd_file   193L_w.crd
output_pdb_file   193L_w.pdb

#モデルビルディング#
alias OXT OT2
alias CD CD1
alias HOH TIP3
alias O OH2
alias O OT1
rename_residue 15A   HSE
patch DISU 6A   127A
patch DISU 30A 115A
patch DISU 64A 80A
patch DISU 76A 94A
patch_ter NTER 1A
patch_ter CTER 129A

#システムビルディング#
solvent_pdb_file watbox216.pdb
solvent_cube on
align_axis diagonal
solvent_buffer 15
ion_placement random
ion SOD CLA
```

molx のインプットファイル中で、行っている内容は以下のとおりです

#力場#

この計算に用いる力場を以下のように設定をしています。

- charm_top_file ../../toppar/top_all27_prot_na.rtf

charmm27 力場の蛋白質・核酸用の top ファイル（top_all27_prot_na.rtf）を指定します

- charm_par_file ../../toppar/par_all27_prot_na.prm
  charmm27 力場の蛋白質・核酸用の par ファイル（par_all27_prot_na.prm）を指定しています。

#インプット#

- input_pdb_file　../../pdbfile/193L.pdb
  計算対象の分子の構造のファイル（193L.pdb)を指定しています。

#アウトプット#

- output_mdat_file 193L_w.mdat
  molx を実行した結果出力される mdat ファイルを 193L_w.mdat と指定します。
- output_crd_file　193L_w.crd
  molx を実行した結果出力される crd ファイルを 193L_w.crd と指定します。
- output_pdb_file　193L_w.pdb
  molx を実行した結果出力される PDB ファイルを 193L_w.pdb と指定しています。

#モデルビルディング#

以下の設定を行っています

- alias コマンドにより、入力で指定した PDB ファイルの中で用いられいる原子名を、charmm 力場で用いられている原子名に変更しています。
- HIS51 の残基名を rename_resitude コマンドで HSE とすることでヒスチジンのプロトン化状態を定義しています。
- patch コマンドを用いて、ジスルフィド結合の定義を行うパッチ DISU を適用しています。
- patch_ter コマンドを用いて、N 末端と C 末端の構造を定義するパッチ NTER, CTER を適用しています。

パッチというのは、charmm 力場の中であらかじめ定義されている、化学修飾（ジスルフィド結合の定義やプロトン化状態のコントロールなど）を行うスクリプトです。どのようなパッチが用意されているのかおよび、その利用方法は charm の top ファイルを参考にしてください。

ここで、rename_residue, patch および patch_ter の各コマンドで、残基の指定をする際に、"patch DISU 6A 127A"のように、残基番号 + chain ID という順番で記述していますが、"patch DISU A6 A127"のように、chain ID + 残基番号という順番での記述も可能です。とくに、残基番号がマイナスになるような場合には、chain ID が先になるように記述してください。

#システムビルディング#

計算を行う箱や溶媒およびイオンのセットアップを以下のプロセスで行っています。

- solvent_pdb_file watbox216.pdb
  発生させる水分子の元の構造を watbox216.pdb の pdb ファイルに指定します。このファイルの中身は、一辺が 18.77Å の立方体の中に、216 分子の水がランダムに配置したものです。このファイルの水の情報を周期的に並べることで、この後に定義される、周期境界の箱の中に水を満たします。
- solvent_cube on
  周期境界の箱を立方体に指定します。
- align_axis diagonal
  周期境界の箱を定義する際の蛋白質の配置方法を指定します。**molx** では蛋白質の座標の XYZ 方向の最大値および最小値の位置から、solvent_buffer という変数で指定される厚みに水を配置することにより、周期境界の箱を定義します。align_axis diagonal を指定することにより、蛋白質の慣性主軸の一番長い方向が、箱の対角線にそろうように指定されます（こうすることで、周期境界の箱のサイズを小さく定義することができます。）
- solvent buffer 15
  周期境界の箱の中心に置かれた蛋白質と箱の各面までの溶媒の厚さを 15Åとします。
- ion_placement random
  イオンの配置を系の中にランダムに行います。
- ion SOD CLA
  アニオンとカチオンにそれぞれ塩素イオン（CLA）、ナトリウムイオン（SOD）を用いていています。このほかのイオンとして、CHARMM のファイルに指定してあるイオンが使用可能です（top ファイルなどを参考にしてください）。（ただし、ここで用いることができるのは、アニオン・カチオンともに一価のイオンのみです）ここでは、系の全電荷をゼロにするために必要最低量のイオンを配置します。

上に示したインプットファイルにより、**molx** を実行すると以下のように出力されます。

```
> molx.0.5.11b molx.in
****************************************
      Molx (Version 0.5.11b)
      Host: bits1
      Date: Wed Aug 22 12:26:17 2012
      Control File: molx.in
****************************************

CHARMM TOP FILE: ../../toppar/top_all27_prot_na.rtf
  Version 31.1
  Number of types of atomic mass : 158
  Number of residues : 37
  Number of residues for patching : 31
```

```
CHARMM PAR FILE: ../../toppar/par_all27_prot_na.prm
   Number of bond types:    257
   Number of angle types: 656
   Number of dihedral types: 1127
   Number of improper dihedral types: 70
   Number of cmap dihedral types: 6
   Number of nonbonded atom types: 158
   Number of modified nonbonded atom pairs: 0

PDB FILE:    ../../pdbfile/193L.pdb
   Number of atoms : 1001
   Number of hetero atoms :    144
   Number of residues : 273


Renaming residue     15A HIS -> HSE
Warning: CMAP term not assigned:    -C N CA C N CA C +N in residue 1A
Warning: CMAP term not assigned:    -C N CA C N CA C +N in residue 129A
Patching DISU to CYS(6A) and CYS(127A)
Patching DISU to CYS(30A) and CYS(115A)
Patching DISU to CYS(64A) and CYS(80A)
Patching DISU to CYS(76A) and CYS(94A)
Patching NTER to LYS(1A)
Patching CTER to LEU(129A)
1243 atoms are missing in pdb file.

Coordinates of hydrogens of 142 crystal waters are generated.
All Coordinates are determined.

142 waters are found in input pdb file.
1 cations are found in input pdb file.
1 anions are found in input pdb file.

Align principal axes of molecules.
Align the longest axis to diagonal of a cube
Setup of Solvation.
Minimum and maximum coordinates of solute:
   (-19.13,-16.18,-22.41)-(19.67,17.41,20.26)
solvent_buffer 15.00 Angstrom
solvent_cube option: on
All atoms are shifted: (36.06,35.72,37.41)
Simulation box is configured as (72.66,72.66,72.66)
Solvent PDB file:
PDB FILE:    watbox216.pdb
   Number of atoms : 648
   Number of hetero atoms :    0
   Number of residues : 216
Box Size of Input Solvent PDB: (18.77,18.77,18.77)
Duplicated: (4,4,4)
solvent_radius 1.40 Angstrom
solvent_exclusion_layer 0.00 Angstrom
ION: Grid Spacing     (1.00,1.00,1.00)
ION: Number of Grid (73,73,73)
ION: placement random
ION: ion_cutoff 7.40 Angstrom
ION: solvent_radius 1.40 Angstrom
ION: ion_exclusion_layer 4.00 Angstrom
ION: ion2_exclusion_layer 2.00 Angstrom
ION: Starting to Charge Grid    Done
```

```
ION: SOD 0, CLA 8
ION: anion    CLA (10.09,10.13,9.55) by random
ION: anion    CLA (14.35,3.11,6.71) by random
ION: anion    CLA (12.82,16.72,43.93) by random
ION: anion    CLA (16.77,45.42,33.76) by random
ION: anion    CLA (9.53,31.58,28.33) by random
ION: anion    CLA (4.50,22.08,30.62) by random
ION: anion    CLA (33.41,62.03,52.86) by random
ION: anion    CLA (38.16,15.64,8.34) by random
ION: SOD 0, CLA 8

PDB FILE:    configured_solvent
   Number of atoms : 34878
   Number of hetero atoms :    0
   Number of residues : 11626

0 atoms are missing in pdb file.
Molecular Data (mdat) Information:
   Number of atoms: 37274
   Number of atom types: 37
   Number of residues: 11907
   Number of molecules: 11779
   Number of bonds: 37288
   Number of bond types: 69
   Number of angles: 15315
   Number of angle types: 151
   Number of dihedrals: 5187 (term: 5391)
   Number of dihedral types: 185
   Number of impropers: 351
   Number of improper types: 14
   Number of cmap terms: 127
   Number of cmap types: 4
Number of solute molecules: 1
Total charge: -0.000000
Periodic Boundary Box:
72.66     0.00     0.00
0.00    72.66     0.00
0.00     0.00    72.66
```

3.2.3 molx 計算例ー２ ：F1 モーター

ここでは F1 モーターの結晶構造(PDB_ID:2JBI)における、2 量体(chain B と F)について、**molx** を実行する場合の例を示します。

3.2.3.1 molx を実行する前に

まず、この前の章の"**3.2.1 molx を実行する前に**"に示した、チェック項目(a)～(d)について確認します。

(a)蛋白質分子の原子欠損の有無

2JBI の計算に用いる chain B と F について、以下の残基番号の領域のすべての原子の位置の情報が欠落しています。

chain B : 残基番号 1 から 22, 402 から 409

chain F: 残基番号-4 から 9, 475 から 478

これらのうち、chain B の 402 から 409 は、chain B の蛋白質の途中の部分であるために、この領域の構造情報が存在しないと、chain B が分断されてしまうため、この領域について、プログラム modeller によってモデリングした座標を構築しました。それ以外の領域はいずれも、N 末端および C 末端であり、欠損していても機能上影響がないものと判断しました。

(b)マルチコンフォマーの有無

2JBI の chain B と F の構造にはマルチコンフォマーは存在しません。

(c) アミノ酸のプロトン化状態の決定

存在する荷電アミノ酸はすべて電荷のある状態に定義します。さらに、ヒスチジンについては、"**molx** を実行する前に"に示したように、3 つの状態があり、そのうち HSD と HSE は側鎖の電荷が中性です。2JBI の構造では chain B に 5 つ（残基番号 42, 263, 302, 471, 476), chain F に 8 つ（残基番号 52, 117, 177, 198, 328, 367, 427, 451)のヒスチジンが存在しますが、以下のようにプロトン化状態を定義します。

chain B : 42(HSD), 263(HSD), 302(HSE), 471(HSD), 476(HSD)

chain F : 52(HSE), 117(HSE), 177(HSE), 198(HSD), 328(HSD), 367(HSE), 427(HSD), 451(HSE)

(d) ジスルフィド結合の有無

2JBI には、ジスルフィド結合は存在しません。

3.2.3.2 molx の実行

以上のことを考慮して、**molx** を実行するためのインプットファイルを作成します。
以下に、F1 モーターの X 線構造（PDB_ID: 2JBI)の chain B,F について、モデリングによる修正を加えた pdb ファイル(2JBI_BFsub.pdb)を用いた水中の系を構築する例 を示します。（以下の例の中で、">"で示されているのは、コマンドプロンプトです。）

```
> cat molx.in
#力場#
charmm_top_file toppar/top_all27_prot_na.rtf
charmm_par_file toppar/par_all27_prot_na.prm
charmm_toppar_file toppar/stream/toppar_all27_na_nad_ppi.str
charmm_toppar_file toppar_all27_na_po4.str

#入力#
input_pdb_file 2JDI_BFsub.pdb

#出力#
output_mdat_file 2JDI_BFsub_molx.mdat
output_crd_file   2JDI_BFsub_molx.crd
output_pdb_file   2JDI_BFsub_molx.pdb

#モデルビルディング#
alias ANP ATP
alias N3B O3B
alias OXT OT2
alias O OT1
alias CD CD1
alias HOH TIP3
alias WAT TIP3
alias OW OH2
alias O OH2
alias O OT1
alias H1 HT1
alias H2 HT2
alias H3 HT3
alias H HN
alias HG HG1
alias HD11 HD1
alias HD12 HD2
alias HD13 HD3
alias NA SOD
alias CL CLA
rename_residue   42B HSD
rename_residue 263B HSD
rename_residue 263B HSD
rename_residue 302B HSE
rename_residue 471B HSD
rename_residue 476B HSD
rename_residue   52F HSE
rename_residue 117F HSE
rename_residue 177F HSE
rename_residue 198F HSD
rename_residue 328F HSD
rename_residue 367F HSE
rename_residue 427F HSD
```

```
rename_residue 451F HSE

patch_ter NTER 23B
patch_ter CTER 510B
patch_ter NTER 9F
patch_ter CTER 474F

#システムビルディング
solvent_pdb_file watbox216.pdb
solvent_buffer 14
align_axis on
ion SOD CLA
```

molx のインプットファイル中で、行っている内容は以下のとおりです。

#力場#

この計算に用いる力場を以下のように設定をしています。

- charm_top_file toppar/top_all27_prot_na.rtf
  charmm27 力場の蛋白質・核酸用の top ファイル（top_all27_prot_na.rtf）を指定します
- charm_par_file toppar/par_all27_prot_na.prm
  charmm27 力場の蛋白質・核酸用の par ファイル（par_all27_prot_na.prm）を指定しています。
- charm_toppar_file toppar/toppar_all27_na_nad_ppi.str
  charmm27 力場の ATP 用の toppar ファイル（toppar_all27_na_nad_ppi.str）を指定しています。
- charm_toppar_file toppar_all27_na_po4.str
  charmm27 力場のリン酸の toppar(toppar_all27_na_po4.str)を指定しています。

#インプット#

- input_pdb_file 2JDI_BFsub.pdb
  計算対象の分子の構造のファイル（2JDI_BFsub.pdb)を指定しています。

#アウトプット#

- output_mdat_file 2JDI_BFsub_molx.mdat
  molx の実行結果を出力する mdat ファイルを 2JDI_BFsub_molx.mdat に指定します。
- output_crd_file　2JDI_BFsub_molx.crd
  molx の実行結果を出力する crd ファイルを 2JDI_BFsub_molx.crd に指定します。
- output_pdb_file　2JDI_BFsub_molx.pdb
  molx の実行結果を出力する pdb ファイルを 2JDI_BFsub_molx.pdb に指定します。

#モデルビルディング#

以下の設定を行っています

- alias コマンドにより、インプットで指定した PDB ファイルの中で用いられている原子名を、charmm 力場で用いられている原子名に変更しています。
- ヒスチジンの残基名を rename_resitude コマンドで変更することでヒスチジンのプロトン化状態を定義しています。
- patch_ter コマンドを用いて、chain B および F の N 末端と C 末端の構造の定義にそれぞれパッチ NTER, CTER を適用しています。

ここで、rename_residue, patch および patch_ter の各コマンドで、残基の指定をする際に、"patch DISU 6A 127A"のように、残基番号 + chain ID という順番で記述していますが、"patch DISU A6 A127"のように、chain ID + 残基番号という順番での記述も可能です。とくに、残基番号がマイナスになるような場合には、chain ID が先になるように記述してください。

#システムビルディング#

計算を行う箱や溶媒およびイオンのセットアップを以下のプロセスで行っています。

- solvent_pdb_file watbox216.pdb
  発生させる水分子の元の構造を watbox216.pdb の pdb ファイルに指定します。このファイルの中身は、一辺が 18.77Å の立方体の中に、216 分子の水がランダムに配置したものです。このファイルの水の情報を周期的に並べることで、この後に定義される、周期境界の箱の中に水を満たします。
- solvent_cube on
  周期境界の箱を立方体に指定します。
- align_axis diagonal
  周期境界の箱を定義する際の蛋白質の配置方法を指定します。**molx** では蛋白質の座標の XYZ 方向の最大値および最小値の位置から、solvent_buffer という変数で指定される厚みに水を配置することにより、周期境界の箱を定義します。align_axis diagonal を指定することにより、蛋白質の慣性主軸の一番長い方向が、箱の対角線にそろうように指定されます（こうすることで、周期境界の箱のサイズを小さく定義することができます。）
- solvent buffer 15
  周期境界の箱の中心に置かれた蛋白質と箱の各面までの溶媒の厚さを 15Åとします。
- ion_placement random
  イオンの配置を系の中にランダムに行います。
- ion SOD CLA
  アニオンとカチオンにそれぞれ塩素イオン（CLA）、ナトリウムイオン（SOD）を用いていています。このほかのイオンとして、CHARMM のファイルに指定

してあるイオンが使用可能です。（ただし、ここで用いることができるのは、アニオン・カチオンともに一価のイオンのみです）

- ion_density 150
  ここでは、系の全電荷をゼロにするためにアニオンとカチオンにそれぞれ塩素イオン（CLA)とナトリウムイオン（SOD)をランダムに配置していますが、ion_density 150 を指定することで、生理的な塩濃度を考慮イオン濃度（150mM）になるように、イオンを発生させて配置します。

上に示したインプットファイルで **molx** を実行すると以下のように出力されます。

```
> molx.0.5.11b molx.in
****************************************
      Molx (Version 0.5.11b)
      Host: bits1
      Date: Wed Aug 29 18:28:01 2012
      Control File: molx.in
****************************************

CHARMM TOP FILE: toppar/top_all27_prot_na.rtf
  Version 31.1
  Number of types of atomic mass : 163
  Number of residues : 37
  Number of residues for patching : 31

CHARMM PAR FILE: toppar/par_all27_prot_na.prm
  Number of bond types:   257
  Number of angle types: 656
  Number of dihedral types: 1127
  Number of improper dihedral types: 70
  Number of cmap dihedral types: 6
  Number of nonbonded atom types: 163
  Number of modified nonbonded atom pairs: 0

CHARMM TOP FILE: + toppar/stream/toppar_all27_na_nad_ppi.str
  Version 31.1
  Number of types of atomic mass : 163
  Number of residues : 48
  Number of residues for patching : 32

CHARMM PAR FILE: + toppar/stream/toppar_all27_na_nad_ppi.str
  Number of bond types:   257
  Number of angle types: 656
  Number of dihedral types: 1127
  Number of improper dihedral types: 70
  Number of cmap dihedral types: 6
  Number of nonbonded atom types: 163
  Number of modified nonbonded atom pairs: 0

CHARMM TOP FILE: + toppar_all27_na_po4.str
  Version 31.1
  Number of types of atomic mass : 163
  Number of residues : 49
  Number of residues for patching : 32
```

```
CHARMM PAR FILE: + toppar_all27_na_po4.str
   Number of bond types:   257
   Number of angle types: 656
   Number of dihedral types: 1127
   Number of improper dihedral types: 70
   Number of cmap dihedral types: 6
   Number of nonbonded atom types: 163
   Number of modified nonbonded atom pairs: 0

PDB FILE:   2JDI_BFsub.pdb
   Number of atoms : 7252
   Number of hetero atoms :   64
   Number of residues : 958

Renaming residue     42B HIS -> HSD
Renaming residue    263B HIS -> HSD
Renaming residue    302B HIS -> HSE
Renaming residue    471B HIS -> HSD
Renaming residue    476B HIS -> HSD
Renaming residue     52F HIS -> HSE
Renaming residue    117F HIS -> HSE
Renaming residue    177F HIS -> HSE
Renaming residue    198F HIS -> HSD
Renaming residue    328F HIS -> HSD
Renaming residue    367F HIS -> HSE
Renaming residue    427F HIS -> HSD
Renaming residue    451F HIS -> HSE
Warning: CMAP term not assigned:   -C N CA C N CA C +N in residue 23B
Warning: CMAP term not assigned:   -C N CA C N CA C +N in residue 510B
Warning: CMAP term not assigned:   -C N CA C N CA C +N in residue 9F
Warning: CMAP term not assigned:   -C N CA C N CA C +N in residue 474F
Patching NTER to VAL(23B)
Patching NTER to THR(9F)
Patching CTER to ALA(510B)
Patching CTER to ALA(474F)
Warning: Atom OT2   in residue ALA ( 510B) is missing in pdb file.
Warning: Atom OT2   in residue ALA ( 474F) is missing in pdb file.
7443 atoms are missing in pdb file.

All Coordinates are determined.

0 waters are found in input pdb file.
0 cations are found in input pdb file.
0 anions are found in input pdb file.
Align principal axes of molecules.
Setup of Solvation.
Minimum and maximum coordinates of solute:
   (-47.15,-42.56,-30.51)-(49.19,38.09,34.75)
solvent_buffer 14.00 Angstrom
All atoms are shifted: (61.15,56.56,44.51)
Simulation box is configured as (124.34,108.65,93.26)
Solvent PDB file:
PDB FILE:   watbox216.pdb
   Number of atoms : 648
   Number of hetero atoms :   0
   Number of residues : 216
```

```
Box Size of Input Solvent PDB: (18.77,18.77,18.77)
Duplicated: (7,6,5)
solvent_radius 1.40 Angstrom
solvent_exclusion_layer 0.00 Angstrom
ION: Grid Spacing     (0.99,1.00,0.99)
ION: Number of Grid (125,109,94)
ION: placement random
ION: ion_cutoff 9.40 Angstrom
ION: solvent_radius 1.40 Angstrom
ION: ion_exclusion_layer 6.00 Angstrom
ION: ion2_exclusion_layer 2.00 Angstrom
ION: Starting to Charge Grid    Done
ION: total charge is -18
ION: 35740 waters are solvated.
ION: 0 ions are in input pdb file
ION: Number of ions is estimated to be 194.
ION: 194 ions are added.
ION: SOD 106, CLA 88
ION: cation SOD (10.09,10.13,9.55) by random
ION: cation SOD (4.89,6.16,11.47) by random
ION: cation SOD (26.59,9.98,5.39) by random
……

途中略

……
ION: cation SOD (22.49,19.10,43.41) by random
ION: anion    CLA (8.46,20.02,59.78) by random
ION: cation SOD (25.23,35.22,9.99) by random
ION: anion    CLA (117.29,29.15,19.92) by random
ION: SOD 106, CLA 88
PDB FILE:    configured_solvent
   Number of atoms : 106638
   Number of hetero atoms :    0
   Number of residues : 35546

0 atoms are missing in pdb file.

Molecular Data (mdat) Information:
   Number of atoms: 121591
   Number of atom types: 56
   Number of residues: 36698
   Number of molecules: 35746
   Number of bonds: 121507
   Number of bond types: 90
   Number of angles: 62545
   Number of angle types: 196
   Number of dihedrals: 39318 (term: 41017)
   Number of dihedral types: 295
   Number of impropers: 2427
   Number of improper types: 18
   Number of cmap terms: 950
   Number of cmap types: 6
   Number of solute molecules: 6
   Total charge: -0.000000
   Periodic Boundary Box:
         124.34      0.00      0.00
           0.00 108.65      0.00
           0.00      0.00    93.26
```

3.3 MARBLE

3.3.1 エネルギー最小化（リゾチームを例に）

MARBLE を用いて計算対象の系のエネルギー最小化を行います。 このエネルギー最小化は、**molx** で発生させた欠損した水素原子、および周辺に発生させた溶媒分子の構造最適化を行うことが目的です。以下にインプットファイルを示します。 ここでは、最急降下法で 1500 ステップのエネルギー最小化計算を行います。この計算の間、リゾチームの水素以外の原子はすべて 193L_w.crd の位置に拘束をかけています。

```
> more min.in
[input]
 mdat_file = ../molx/193L_w.mdat
 crd_file   = ../molx/193L_w.crd

[nonbond]
 cutoff = 10.0

[ewald]
 d_grid = 1.1

[restraint]
 method = position_harmonic
 crd_file = ../molx/193L_w.crd
 group = atom non_hydrogen 1A 129A
 k = 1.0 #kcal/mol/ang2

[min]
 step = 1500

[output]
 crd_file = 193L_w-min.crd
 pdb_file = 193L_w-min.pdb
```

MARBLE のインプットファイルでの設定は、[ ]で示されている、各セクションで行います。 エネルギー最小化計算のインプットファイルでの設定は、以下のとおりです。

[input]

インプットファイルとして以下のファイルを設定します。

- mdat_file = ../**molx**/193L_w.mdat
  mdata ファイルとして 193L_w.mdat を指定します。
- crd_file = ../**molx**/193L_w.crd
  crd ファイルとして 193L_w.crd を指定します。

[nonbond]

非結合相互作用の設定を行います。

- cutoff = 10.0
  cutoff で近距離相互作用のカットオフを 10Å に設定しています。

[ewald]

非結合長距離相互作用の計算方法である PME (Particle Mesh Ewald)の設定を行います。

- d_grid = 1.1
  PME を行う際に、周期境界の箱の中に定義するグリッドの間隔の上限値を定義します。値は 1.1 程度にしてください。

[restraint]

特定の各原子グループに対して、position harmonic により、1 (kcal·mol^{-1}Å^{-2})の力の定数で 193L_w.crd の位置に拘束をかける設定を以下のようにしています。

- method = position_harmonic
  拘束をかける方法を、position harmonic に指定します。この方法は、拘束をかけたい原子について、指定した座標の位置と、シミュレーション中の座標の間をばねで結んで拘束をかけます。
- crd_file = ../**molx**/193L_w.crd
  拘束をかけるために指定する座標の位置を含む crd ファイルを指定します。ここでは、193L_w.crd に指定しています。
- group = atom non_hydrogen 1A 129A
  group コマンドで指定したグループに含まれる原子グループを定義します。ここでは、chain ID が A の分子の残基番号が 1 から 129 までの残基の heavy atom（水素原子以外の原子）を指定しています。
- k = 1.0
  拘束に用いるばねのばね定数を指定します。ここでは、1.0 (kcal·mol^{-1}Å^{-2})に指定しています。

[min]

このセクションでは、エネルギー最小化の設定を行います。

- step = 1500
  1500 ステップの計算を行うことを設定します。（ここでは再急降下法によるエネルギー最小化が行われます）。

[output]

アウトプットファイルを設定します。

- crd_file = 193L_w-min.crd
  crd ファイルとして 193_w-min.crd を指定します
- pdb_file = 193L_w-min.pdb
  pdb ファイルとして 193_w-min.pdb を指定します。

3.3.2 分子動力学計算

MARBLE による分子動力学計算の例として、**molx** の計算例で示したリゾチームと F1 モーターを用いた以下の計算例について、インプットファイルを元に解説します。

（3.3.2.1）リゾチーム水溶液系の分子動力学シミュレーション
（3.3.2.2）F1 モーターの closed 構造から open 構造への Targeted MD

3.3.2.1 リゾチーム水溶液系の分子動力学シミュレーション

ここでは、エネルギー最小化を行ったリゾチーム水溶液系の座標を用いて **MARBLE** での分子動力学法によるシミュレーションを行うまでのプロセスを示します。計算は以下のプロセスで行います。

平衡化（温度をシミュレーション温度まで上昇させる）
さらに、指定の温度を維持したまま、蛋白質の拘束を外してゆきます。
本計算（Productive run）

ここでは例として **molx** のチュートリアルで作成した、リゾチームの結晶構造 193L の水溶液系の平衡化についての **MARBLE** のインプットファイルを以下に示します。

3.3.2.1.1 平衡化（温度をシミュレーション温度まで上昇させる）

まず、系の温度をシミュレーション温度（ここでは 300K）まで徐々に上昇させます。このシミュレーションのインプットファイルを以下に示します。この計算を行う間、リゾチームの水素以外の原子は、193L_w.crd の位置で拘束をかけておきます。

```
> more eq00.in
[input]
  mdat_file = ../molx/193L_w.mdat
  crd_file    = ../minimize/193L_w-min.crd

[init]
  temperature = 10

[nonbond]
  cutoff = 10.0

[ewald]
  d_grid = 1.1

[PT_control]
  ensemble = NVT
  method = rescaling
  temperature = 10
  gradual_change_T = 20000 300.0

[constraint]
  rigid_body = hydrogen
```

```
[restraint]
method = position_harmonic
crd_file = ../molx/193L_w.crd
group = atom non_hydrogen 1A 129A
k = 1.0

[md]
time_step = 2.0
step = 50000
trj_file = 193L_w-eq00.trj
trj_step = 500
print_step = 100
prop_file = 193L_w-eq00.prop
prop_step = 50

[output]
crd_file = 193L_w-eq00.crd
pdb_file = 193L_w-eq00.pdb
```

MARBLE のインプットファイルでの設定は、[　]で示されている、各セクションで行います。インプットファイルでの指定内容を以下に示します。（前の章で示した計算のインプットファイルと同一内容のものについては、記述のある章を参照してください。）

[init]

- temperature = 10
  系の初期速度を温度 10(K)に設定します。

[PT-control]

- ensemble = NVT
  系のアンサンブルを NVT に設定しています。
- method = rescaling
  温度をコントロールする手法として rescaling 法を指定しています。
- temperature = 10
  温度コントロールの初期温度を 10 (K)とします。
- gradual_change_T = 20000 300.0
  20000 ステップの間に初期温度の 10(K)から 300(K)まで温度を上昇させるように設定します。

[constraint]

- rigid_body = hydrogen
  分子動力学シミュレーションを行う際に、剛体として取り扱う原子を水素原子が共有結合している原子団に設定します。

[md]

このセクションでは、分子動力学シミュレーションの設定を行います。

- time_step = 2.0
  シミュレーションは 1 ステップあたり 2.0 (fs)とします。
- step = 50000
  シミュレーションは全部で 50,000 ステップ（100(ps))行います。
- trj_file = 193L_w-eq00.trj
  トラジェクトリが出力される trj ファイルを 193L_w-eq00.trj とします。
  （trj ファイルには系の全原子の座標と周期境界の箱の情報が出力されます。）
- trj_step = 500
  trj ファイルに出力する頻度を 500 ステップごとに指定します。
- print_step = 100
  out ファイルへのエネルギーなどの出力を 100 ステップごととします。
- prop_file = 193L_w-eq00.prop
  エネルギーなどが出力される prop ファイルを 193L_w-eq00.prop とします。
- prop_step = 50
  prop ファイルに 50 ステップごとに、情報を出力します。

3.3.2.1.2　　平衡化（拘束を徐々に外す）

molx の章で用いたリゾチームの結晶構造 193L の水溶液系の平衡化の後半のシミュレーションを示しています。このシミュレーションでは、この前の”平衡化（温度をシミュレーション温度まで上昇させる）”のシミュレーションの後に、リゾチームの水素以外の原子にかけていた拘束を徐々に外してゆきます。以下に、インプットファイルを示します。

```
> more eq01.in
[input]
  mdat_file = ../molx/193L_w.mdat
  crd_file    = 193L_w-eq00.crd
  restart = on

[nonbond]
  cutoff = 10.0

[ewald]
  d_grid = 1.1

[PT_control]
  ensemble = NVT
  method = rescaling
  temperature = 300

[constraint]
  rigid_body = hydrogen
```

```
[restraint]
  method = position_harmonic
  crd_file = ../molx/193L_w.crd
  group = atom non_hydrogen 1A 129A
  k = 1.0
  gradual_change_k = 50000 0


[md]
  time_step = 2.0
  step = 50000
  print_step = 100
  trj_file = 193L_w-eq01.trj
  trj_step = 500
  prop_file = 193L_w-eq01.prop
  prop_step = 50

[output]
  crd_file = 193L_w-eq01.crd
  pdb_file = 193L_w-eq01.pdb
```

MARBLE のインプットファイルでの設定は、[　]で示されている各セクションで行います。インプットファイルでの指定内容を以下に示します。（前の章で示した計算のインプットファイルと同一内容のものについては、記述のある章を参照してください。）

[input]

- restart = on
  インプットファイルとして指定した crd ファイル（ここでは、193L_w-eq00.crd）に含まれている、速度とアンサンブルの情報を引き継いで計算を行います。

[PT_control]

- temperature = 300
  温度を 300 K で実行しています。初期速度は、前の計算から引き継ぐので、[init] セクションで設定していないことに注意してください。

[restraint]

特定の各原子グループに対して、position harmonic により、1 (kcal·mol^{-1}Å^{-2})の力の定数で 193L_w.crd の位置に拘束をかける設定を以下のようにしています。さらに、position_harmonic の拘束に用いた力の定数を徐々に 0 まで変化させています。

- gradual_change_k = 50000 0
  拘束に用いた力の定数を 1(kcal·mol^{-1}Å^{-2})から、50,000 ステップかけて 0(kcal·mol^{-1}Å^{-2})に変化させています。

3.3.2.1.3　　本計算（NVT アンサンブル）

以下のインプットファイルでは、この前の”**平衡化（拘束を徐々に外す）**”の計算に引き続いて、本計算（Productive run)として、リゾチーム（193L）の水中の系について 1(ns)の計算を行います。

```
> more run01.in
[input]
  mdat_file = ../molx/193L_w.mdat
  crd_file    = ../equil/193L_w-eq01.crd
  restart = on

[nonbond]
  cutoff = 10.0

[ewald]
  d_grid = 1.1

[PT_control]
  ensemble = NVT
  temperature = 300
  method = extended_system
  initialize = on

[constraint]
  rigid_body = hydrogen

[md]
  time_step = 2.0
  step = 5000000
  print_step = 100
  trj_file = 193L_w-run01.trj
  trj_step = 500
  prop_file = 193L_w-run01.prop
  prop_step = 50

[output]
  crd_file = 193L_w-run01.crd
  pdb_file = 193L_w-run01.pdb
```

MARBLE のインプットファイルでの設定は、[　]で示されている各セクションで行います。インプットファイルでの指定内容は、記述のある章を参照してください。

[PT_control]

- method = extended_system
  拡張座標法（nose-hoover 法）で温度コントロールをします。method の指定がないときは extended_system がデフォルトとなります。
- initialize = on
  パラメータを初期化しています。次以降の計算では、この指定を外して crd ファイル中のパラメータを引き継ぎます。

3.2.3.1.3　　本計算（NPT アンサンブル）

以下のインプットファイルでは、この前の”**平衡化（拘束を徐々に外す）**”の計算に引き続いて、本計算（Productive run)として、リゾチーム（193L）の水中の系について 1(ns)の計算を行います。この本計算は、この前の本計算の場合と異なり、アンサンブルを NVT から NPT に変更して計算を行っています。

```
> more run01.in
[input]
  mdat_file = ../molx/193L_w.mdat
  crd_file    = ../equil/193L_w-eq01.crd
  restart = on

[nonbond]
  cutoff = 10.0

[ewald]
  d_grid = 1.1

[PT_control]
  ensemble = NPT
  temperature = 300
  method = extended_system
  initialize = on

[constraint]
  rigid_body = hydrogen

[md]
  time_step = 2.0
  step = 5000000
  print_step = 100
  trj_file = 193L_w-run01.trj
  trj_step = 500
  prop_file = 193L_w-run01.prop
  prop_step = 50

[output]
  crd_file = 193L_w-run01.crd
  pdb_file = 193L_w-run01.pdb
```

MARBLE のインプットファイルでの設定は、[　]で示されている各セクションで行います。インプットファイルでの指定内容は、記述のある章を参照してください。

[PT-control]

- ensemble = NPT
  系のアンサンブルを NPT に設定しています。

- method = extended_system
  拡張座標法で温度・圧力コントロールをします。method の指定がないときは、extended_system がデフォルトになります。
- initialize = on
  NPT 用にパラメータを初期化します。次以降の計算では、この指定を外して、インプットの.crd ファイルに含まれている NPT アンサンブルの情報を引き継ぎます。

3.3.2.2 F1 モーターの closed 構造から open 構造への Targeted MD

Targeted MD とは、蛋白質の原子に対して目標の構造に近づくように力をかけることで、蛋白質が実際構造変化する時間スケールよりはるかに速い速度で蛋白質の構造変化を実現することができます。 ここでは、F1 モーターのβサブユニットの原子に力をかける本計算 (Productive Run)のインプットファイルを示します。

本計算（Production Run）
Targeted MD の本計算のインプットファイル示します。 ここでは平衡化終了後の系の座標を用いて、βサブユニットが closed→open に構造変化する計算を(total 5ns で)行います。

```
>cat run.in
[input]
 mdat_file   =   ../molx/2JDI_BFsub_molx.mdat
 crd_file    = ../eq2/2JDI_BFsub_eq02.crd
 restart = on

[nonbond]
 cutoff = 10.0

[ewald]
 d_grid = 1.1

[constraint]
 rigid_body = hydrogen

[PT_control]
 ensemble = NVT
 temperature = 300

[init]
 solute_molecule = 4

[md]
 time_step = 2.0
 step = 250000
 remove_momentum = solute_rot
 print_step = 500
 trj_file = 2JDI_BFsub_rmsd01_001.trj
 trj_step = 500
 prop_file = 2JDI_BFsub_rmsd01_001.prop
 prop_step = 5000
```

```
[output]
 crd_file     = 2JDI_BFsub_rmsd01_001.crd
 pdb_file     = 2JDI_BFsub_rmsd01_001.pdb

[restraint]
 method = rmsd
 k = 7300.0
 pdb_file = 2JDI_AEsub.pdb
 group = atom non_hydrogen 24B 601B
 group = atom non_hydrogen 9F 474F
 pdb_group = atom non_hydrogen 24B 601B
 pdb_group = atom non_hydrogen 9F 474F
 best_fit = on
 rmsd = 5.62243
 gradual_change_rmsd = 2500000 0.0
```

MARBLE のインプットファイルでの設定は、[　]で示されている各セクションで行います。インプットファイルでの指定内容を以下に示します。（前の章で示した計算のインプットファイルと同一内容のものについては、記述のある章を参照してください。）

[init]

- solute_molecule = 4
  このシミュレーションでは、[md]の項目で remove_momentum = solute_rot と指定することで、溶質分子の回転を止める設定を行いますが、ここに示した solute_molecule コマンドで回転を止める溶質分子の数を指定します。ここでは、溶質分子の数は 4 (F1 の α β サブユニット、ATP および Mg)を指定しています。

[restraint]

MARBLE で Targeted MD を行う際、各原子の目標とする構造に近づく方向へかける力は、到達させたい構造とシミュレーション中の構造の間の RMSD 値を維持するための拘束力として与えます。そして、維持する RMSD 値を目標構造に近づけるように（つまり、RMSD 値が 0 に近づくように）拘束する RMSD 値を変化させてゆくシミュレーションを行うことで、Targeted MD を行っています。

- method = rmsd
  拘束の方法として目標の構造との間の RMSD 値による拘束を指定します。
- k = 7300.0
  RMSD 値での拘束は、シミュレーション中の構造と到達させたい構造の間の各原子の RMSD 値に応じて変化するばねにより拘束しますが、ここでは、系で用いるばね定数のトータルの値を指定します。ここで示している計算例では、拘束をかけるのは、B 鎖の 24～601 番、及び F 鎖の 9～474 番の残基の重原子で、その総数が約 7300 個です。ここでは原子 1 つあたり 1($kcal \cdot mol^{-1} Å^{-2}$)のバネ定数を想定しているので、k は 7300 となります。

- pdb_file = 2JDI_AEsub.pdb
  構造変化させる目標の構造を指定します。ここでは、F1 モーターのβサブユニットが closed→open に変化する計算ですので、構造変化させる目標の構造として 2JDI_AEsub.pdb(open 構造)です。
- group = atom non_hydrogen 24B 601B
  group = atom non_hydrogen 9F 474F
  RMSD 値による拘束をかける原子を指定します（ここでは、インプットファイルで指定した 2JDI_BFsub_eq02.crd に含まれる原子中で拘束をかける原子を指定します。）
- pdb_group = atom non_hydrogen 24B 601B
  pdb_group = atom non_hydrogen 9F 474F
  目標の構造（ここでは、2JDI_AEsub.pdb）における RMSD 値を計算するために用いる原子を指定します。
- best_fit = on
  RMSD 値の計算を行う際に、シミュレーション中の構造と目標の構造の重なりを best fit を用いて修正することを指定します（これは、RMSD の計算のために行う操作で、シミュレーション中の構造は、影響を受けません）。
- rmsd = 5.62243
  拘束する RMSD 値を指定します。（ここでは、計算する前の closed 構造と目標の open 構造との間の RMSD 値 5.62243Å を指定します）
- gradual_change_rmsd = 2500000 0.0
  拘束する RMSD の値を 2,500,000 ステップかけて目標構造と重なるように(RMSD=0 となるよう)に指定します。

prop ファイル

上記のインプットファイルで Targeted MD を行った時に出力された prop ファイルの中身を以下に示します。

```
#1 TIME    2 TEMPERATURE    3 TOTAL_ENE    4 POTENTIAL    5 RMSD_ENE    6 RMSD    7 TARGET_RMSD
  2.100000e+02   3.013549e+02 -3.196633e+05 -3.941214e+05   9.339044e-03   5.610054e+00   5.611185e+00
  2.200000e+02   3.004271e+02 -3.196564e+05 -3.936027e+05   2.758408e-02   5.601884e+00   5.599940e+00
  2.300000e+02   2.992376e+02 -3.196625e+05 -3.940014e+05   1.615850e-01   5.583991e+00   5.588695e+00
  2.400000e+02   2.998921e+02 -3.196609e+05 -3.934682e+05   3.922606e-03   5.576718e+00   5.577451e+00
  2.500000e+02   3.001248e+02 -3.196620e+05 -3.936488e+05   7.254864e-01   5.556237e+00   5.566206e+00
  2.600000e+02   3.005848e+02 -3.196579e+05 -3.933531e+05   5.966773e-01   5.564002e+00   5.554961e+00
  2.700000e+02   2.998154e+02 -3.196666e+05 -3.942913e+05   3.877468e-04   5.543486e+00   5.543716e+00
  2.800000e+02   3.011331e+02 -3.196645e+05 -3.937764e+05   3.908584e-02   5.530157e+00   5.532471e+00
  2.900000e+02   2.993740e+02 -3.196623e+05 -3.938691e+05   4.938038e-01   5.513002e+00   5.521226e+00
  3.000000e+02   2.997662e+02 -3.196665e+05 -3.941785e+05   4.499579e-01   5.502130e+00   5.509981e+00
-------------
途中略
-------------
  5.160000e+03   3.012046e+02 -3.161688e+05 -3.924416e+05   1.368356e+03   4.779299e-01   4.497944e-02
  5.170000e+03   3.007494e+02 -3.160989e+05 -3.927360e+05   1.435828e+03   4.772306e-01   3.373458e-02
  5.180000e+03   2.995500e+02 -3.160180e+05 -3.922802e+05   1.462634e+03   4.701066e-01   2.248972e-02
  5.190000e+03   2.998215e+02 -3.159517e+05 -3.928270e+05   1.523464e+03   4.680749e-01   1.124486e-02
  5.200000e+03   3.000127e+02 -3.158740e+05 -3.925378e+05   1.564234e+03   4.629024e-01   0.000000e+00
```

カラムの値は左から、計算時間、温度、total エネルギー、potential エネルギー、拘束にかかったエネルギー、現時点での RMSD 値、目標とする RMSD 値を示します。6 番目のカラムの RMSD 値が、初期値:5.62243 から目標値:0.0 に近づいてきていることが分かります。（初期値と目標値は前述のインプットファイルの[restraint]のセクションを参照）

4　molx, MARBLE の実行方法（京, FX10 の場合）

京や FX10 では、molx および MARBLE は計算ノード用の実行ファイルとして生成されます。ここでは、FX10 での計算ノードでの molx および MARBLE の実行方法を示します。

4.1　molx の実行方法

molx はログインノードで実行できないので､計算ノードで実行する必要があります。バッチモードで実行するか、インタラクティブモードで計算ノードにログインした後に実行してください。

4.2　MARBLE の実行方法

ここでは、“3.1.5 MARBLE の並列化計算について“で述べた並列化計算において必要な、系のセル分割、利用するプロセス数、プロセスの 3 次元指定および PME 法で用いるグリッドの指定を行う 2 つの方法での計算の実行方法を示します。（具体例として、FX10 での使用方法を示します）

“3.1.5 MARBLE の並列化計算について“で述べたように、MARBLE で計算を行う際には、次の情報の指定が必要になります。

(a)系の XYZ 方向のセルの分割数
(b)利用するプロセスの数
(c)プロセスの XYZ 方向の配置の数
(d)PME(Particle Mesh Ewald)法による静電相互作用の計算の際に、空間を等間隔に分割したグリッドの XYZ 方向の数”

そして、(a)～(d)の情報は、以下に示すような特定のルールを満たすように決定する必要があります。

プロセスの XYZ 方向の配置の数は、その積が利用するプロセスの数になります。

セルの分割数を決める際に、以下のことを満たすように決めます。

- セルの最小幅は、[nonbond]で指定した cutoff の値に対して、(cutoff + 4.5)/2 となります。(cutoff が 9 だと 6.75; cutoff が 10 だと 7.25)したがって、この最小幅よりも大きな幅で XYZ 各方向のセルの分割数を決めます。
- 各方向のセルの分割数は(c)の各方向のプロセスの配置の数で割り切れる値であること。

PME のグリッドの XYZ 方向の数は以下のことを満たすように決めます。

- グリッドの間隔が約 1.1Å以下であること。

- 各方向のグリッドの数が(c)の各方向のプロセスの配置の数で割り切れる値であること。

MARBLE では、(a)～(d)の情報を与えて計算を実行する方法として次の 2 つの方法が用意されています。以下に、それぞれの方法での計算方法を示します。

4.2.1 d_grid による方法

d_grid を用いると、並列計算を実行する際のプロセス数を設定するだけで、非常に簡単に並列化計算に必要な情報を設定することができます。

まず、バッチジョブを実行する際に必要なシェルスクリプトファイル（batch.sh）を以下のように記述します。

```
#!/bin/sh
#PJM -L "rscgrp=debug"
#PJM -L "node=4x2x2"
#PJM - -mpi “proc=64”
#PJM -L "elapse=30:00"
#PJM -j


export OMP_NUM_THREADS=4
mpiexec /home/xxxx/marble-0.6/bin/marble.0.6.0_FX10 run01.in run01.out
```

上述のシェルスクリプトファイルでは、以下の設定のハイブリッド並列計算です。

ノード数：4x2x2 (#PJM –L “node=4x2x2”)
プロセス数：64 (#PJM - -mpi “proc=64”)
スレッド数：4 (export OMP_NUM_THREADS=4)

この場合、各ノードには、4 プロセスずつ入ることになります。4 プロセスは x, y, z 方向のプロセス数が均等になるように割り振られます。

また、MARBLE を実行するためのインプットファイル（run01.in）は、3 章のインプットファイルで示したように、[ewald]のセクションに、

```
[ewald]
 d_grid = 1.1
```

と、記述することにより、系のボックスサイズから、d_grid の値に基づき PME のグリッドの各方向の数を設定し、使用するプロセス数の情報と合わせて、自動的にセルの各方向の数、プロセスの各方向の数を算出して、計算を実行します。
ただし、この方法は、計算を実行する際に d_grid とボックスのサイズの情報から算出して計算を実行することから、NPT アンサンブルのようなボックスのサイズが変化する条件では、PME のグリッドサイズが途中変更される可能性があります。

4.2.2 直接情報を指定する方法

(a)～(d)の条件を設定するもうひとつの方法は、上述の(A)～(B)の条件を満たすようにこれらの情報を直接インプットファイルで指定する方法です。
まず、バッチジョブを実行する際に必要なシェルスクリプトファイル（batch.sh）を以下のように記述します。

```
#!/bin/sh


#PJM -L "rscgrp=debug"
#PJM -L "node=2x2x4"
#PJM --mpi "proc=64"
#PJM -L "elapse=30:00"
#PJM -j
export MBL_PE_NODE=2x2x1
export OMP_NUM_THREADS=4
mpiexec /home/c74000/marble-0.6/bin/marble.0.6.0_FX10 run01a.in run01a.out
```

上述のシェルスクリプトファイルでは、以下の設定のハイブリッド並列計算です。

ノード数：16 (#PJM –L “node=2x2x4”)
プロセス数：64 (#PJM - -mpi “proc=64”)
スレッド数：4(export OMP_NUM_THREADS=4)

ここで、ノード 1 つあたりのプロセスの数は 4 つですが、通常なにも指定しないと、4 プロセスは x, y, z 方向のプロセス数が均等になるように割り振られます。

自動的な割り振りと異なる割り振りを明示的に指定したい場合、上のシェルスクリプトファイル内の、
export MBL_PE_NODE=2x2x1

を指定してやることで、各ノード内のプロセスの 3 次元分割を 2x2x1 に指定することができます。以上の設定から、ノード、およびノード内のプロセスの 3 次元分割が、それぞれ、2x2x4 および 2x2x1 となるので、計算に用いるすべてのプロセスの 3 次元分割は、4x4x4 となります。

また、MARBLE を実行するためのインプットファイル（run01a.in）の必要な記述箇所だけを示します。

```
……
 [nonbond]
 cutoff = 10.0
 n_cell = 8 8 8
 n_pe = 4 4 4

[ewald]
 method=PME
 grid=72 72 72
……
```

ここで、
[nonbond]のセクションでは、
cutoff = 10.0 を指定しています。
（したがって、セルの最小幅は、上述の計算式より 6.75 となります）
ここで、リゾチームのボックスのサイズは、各辺 72.66Å です（3.2.2 の molx 実行後のアウトプット参照）。したがって上述の(A)～(C)の条件を満たすセル、プロセス、PME のグリッドの XYZ 方向の分割数として、それぞれ、8x8x8, 4x4x4, 72x72x72 と設定しました。

5　コマンドリファレンス

molx および MARBLE で使用するコマンドのリファレンスを示します。各コマンドは太字で示してあります。

5.1　molx

使い方（3.1.1 参照）
molx　*インプットファイル*

使用例
molx　*molx.in*

実際に計算に用いるインプットファイルの例は　“**3.2 molx による系の構築**“を参照のこと。

5.1.1 力場

CHARMM 力場を使用する上で必要となるファイル（top ファイル、par ファイルおよび toppar ファイル）を指定します。

charmm_top_file
CHARMM の top ファイルを指定します(デフォルト値：なし）。top ファイルは複数（最大５つまで）指定することができます。
使用法：　charmm_top_file　*ファイル名*
使用例：　charmm_top_file　　top_all27_prot_na.rtf

charmm_par_file
CHARMM の par ファイルを指定します(デフォルト値：なし）。par ファイルは複数（最大５つまで）指定することができます。
使用法：　charmm_par_file　*ファイル名*
使用例：　charmm_par_file　　par_all27_prot_na.rtf

charmm_toppar_file
CHARMM の toppar ファイルを指定します（デフォルト値：なし）。toppar ファイルは複数（最大５つまで）指定することができます。
使用法：　charmm_toppar_file　*ファイル名*
使用例：　charmm_toppar_file　　toppar_all22_prot_pyridines.str

5.1.2 インプット

計算対象の分子の構造情報を入力情報として指定する。主に pdb ファイルを用いて指定するが、その他に、蛋白質一次構造（アミノ酸配列）を用いて、直鎖ペプチドの計算も可能です。

input_pdb_file

計算に用いる蛋白質などの pdb ファイルを指定する（デフォルト値：なし）。

使用法： input_pdb_file *ファイル名*

使用例： input_pdb_file 6lYZ.pdb

5.1.3 アウトプット

molx で構築したシステムについて marble でシミュレーションを行うために必要なファイル（crd ファイル、mdat ファイル）および、構築したシステムの pdb ファイルを出力します。

output_mdat_file

molx で構築したシステムに関する mdat ファイルを指定します(デフォルト値：なし）。

使用法：output_mdat_file *ファイル名*

使用例：output_mdat_file 6lyz_w.mdat

output_crd_file

molx で構築したシステムに関する crd ファイルを指定します(デフォルト値:なし)。

使用法：output_crd_file *ファイル名*

使用例：output_crd_file 6lyz_w.crd

output_pdb_file

molx で構築したシステム全体の pdb ファイルを指定します(デフォルト値:なし)。

使用法：output_pdb_file *ファイル名*

使用例： output_pdb_file 6lyz_w.pdb

5.1.4 モデルビルディング

計算対象の分子のセットアップを行います。

renumber_residue

残基番号を振り直したいときに指定します。(デフォルト値：)

使用法： renumber_residue {on|off}

使用例： renumber_residue on

rename_residue

インプットファイル中の特定の残基について、その残基名を変更します。（デフォルト値：なし？）

使用法：renumber_residue　*残基番号+chain_ID*　*新しい残基名*

使用例：rename_residue　15A　HSE

bond_length_limit

インプットファイル中の原子間結合距離の上限を指定します。(デフォルト値:？？)

使用法：bond_length_limit　結合距離の上限

使用例：bond_length_limit　5

patch_ter

インプットファイルの蛋白質の N 末端と C 末端を CHARMM 力場で定義されているいくつかの末端を定義する patch で修飾する。

使用法：patch　*patch 名　残基番号(+chain_ID)*

使用例：patch_ter　NTER　23B

patch

インプットファイルの蛋白質を、CHARMM 力場で定義されているいくつかの patch を用いて修飾する場合に用いる。

使用法：patch　*patch 名　残基番号 1 (+chain_ID)　……*

使用例：patch DISU 64A 80A

注意

patch コマンドの使用法は、patch の種類によって、指定する残基の数などは異なるので、CHARMM 力場の patch の情報を参考にしてください。

alias

インプットファイル中の原子名や残基名を変更する際に利用する。

（rename_residue と異なり、インプットファイル中に見られるすべての原子名や残基名に適用される）

使用法：alias *古い名前　新しい名前*

使用例：alias CD CD1

5.1.5 システムビルディング

計算対象の分子の存在する溶媒環境を設定します。このセクションの設定の流れは、ボックスの作成→水の追加→イオンの追加の順番で行われます。

ボックスの作成

MARBLE では、溶媒環境での計算を周期境界条件を用いて行います。したがって、溶媒環境下での計算を行うためには、最初に周期境界を指定するボックスを定義します。

box

ボックスを手動で定義する場合に用います。（ボックスのサイズが決まっている場合にはこのコマンドを用います（デフォルト値：なし。ただし、α、β、γ、については、指定がなければ、90 度に定義されます）

使用法： box *x y z α β γ*

使用例： box 169.10 169.10 170.27

align_axis

分子をボックスに対して指定した方向に揃えます。分子を揃える方向は、"on"、"z"および"diagonal"の 3 つのオプションが用意されています（デフォルト値：なし）。各オプションは、以下のとおり

"on"：　分子の最も長い慣性主軸の方向がボックスの x に揃うように配置。

"z"：分子の最も長い慣性主軸の方向が z 軸に揃うように配置。

"diagonal"：　分子は最も長い慣性主軸の方向がボックスの対角線に合うように配置されます。

使用法： align_axis {normal|z|diagonal}

使用例：align_axis diagonal

solvent_cube

ボックスの形状を立方体にするオプション。"on"の場合、solvent_buffer で定義される溶質表面からの距離から、変の長さを決定し、立方体のボックスを定義する（デフォルト値：なし）。

使用法：solvent_cube　on

使用例：solvent_cube　on

水の追加

溶媒を計算するボックスを定義した後に、計算に使用する溶媒の水分子をボックスの中に発生させます。

solvent_pdb_file

ボックス中に配置する溶媒分子の pdb ファイルを指定します。通常は、wat216.pdb を用います。（デフォルト値：なし）

使用法：solvent_pdb_file *ファイル名*

使用例：solvent_pdb_file　wat216.pdb

solvent_excluded_layer

溶質分子表面の溶媒を排除する層の厚さを定義します（デフォルト値：0Å)。

使用法：solvent_excluded_layer *層の厚さ（Å）*

使用例：solvent_excluded_layer　2

solvent_buffer

溶質からボックスの壁までの層の厚さの最小値を指定します（デフォルト値：10Å)

使用法：solvent_buffer　*溶媒分子の層の厚さ（Å）*

使用例：solvent_buffer　15

イオンの追加

水を発生させた後で、系のトータルチャージがゼロになるようにイオンを発生させます。イオンは、水分子の座標を置き換えることで発生させます。

ion

系に追加するイオンの種類を設定します。（デフォルト値：なし）

使用法： ion　*カチオンの名前　アニオンの名前*

使用例： ion　SOD　CLA

注意

ここで用いているカチオンとアニオンの名前は、CHARMM 力場の中で用いられているイオンの名前です。カチオンとアニオンの名前は正確に、この順番で記述してください。間違えると、molx がイオンを間違ったまま判断してしまい total charge が大きく間違えた値になる恐れがあるので注意してください。ここで用いることができるイオンはカチオン、アニオンともに一価のイオンに限られます。

ion_placement

イオンの配置方法を選択します。イオンは、ランダムに発生させる("random") か、エネルギーが最小になるような位置に発生させる("energy")のどちらかを選択できます。（デフォルト：random）

使用法：ion_placement　{random|energy}

使用例：ion_placement energy

ion_cutoff

イオンのカットオフ距離を設定します。ion_placement が "energy"のときに用います。（デフォルト値：10Å)

使用法：ion_cutoff　カットオフ距離（Å)

使用例：ion_cutoff　7

ion_density

ion コマンドで指定したイオン種を指定したイオン強度になるように、イオンを発生させる。(デフォルト値：0mM)

使用法：　ion_density　*イオン強度(mM)*

使用例：ion_density　4

5.2　MARBLE

使い方（3.1.3 参照）
marble インプットファイル　アウトプットファイル

使用法
marble　run.in　　run.out

実際に計算に用いるインプットファイルの例は　“**3.3 MARBLE**“を参照のこと。

5.2.1 [input]

MARBLE での計算に用いるインプットファイルを指定します。

mdat_file

molx で作成された.mdat ファイル情報を読み込みます。

使用法：　mdat_file　= *ファイル名*
使用例：　mdat_file = ../molx/protein.mdat

crd_file

.crd ファイルから各原子の xyz 座標情報を読み込みます。

使用法：　crd_file　= *ファイル名*
使用例：　crd_file = run001.crd

restart

分子シミュレーションをいくつかのインプットファイルで分割して計算する場合、直前までの計算のアンサンブルや速度の情報を引き継ぐ場合に使います。以下のキーワードを用いて、.crd ファイルの引き継ぐ情報を指定します。
on：crd のすべての情報を引き継いで計算を行います。（デフォルト値：off）
V：速度
B：周期境界のボックスの情報
E: エネルギー
とくに、アンサンブルを変更して計算を始めるときに、V, B, E のキーワードの組み合わせで、以前の計算で得られた情報を選択して、以下に示す[PT_control]のセクションでアンサンブルを指定することで、アンサンブルを変更することができます。（この操作は、restart コマンドを使わないで、[PT_control]で”initialize=on”とすることと、同じです）

使用法：　restart　= {on | off | V | B | E}
使用法：　restart　= VB

5.2.2 [output]

MARBLE での計算に用いるアウトプットファイルを指定します。

crd_file

計算した最終構造の.crd ファイルを出力します。

使用法： crd_file = *ファイル名*

使用例： crd_file = run001.crd

pdb_file

計算した最終構造の.pdb ファイルを出力します。

使用法： pdb_file = *ファイル名*

使用例： pdb_file = run001.pdb

5.2.3 [init]

このセクションでは、計算の設定の初期化（initialize）に関する設定を行います。

temperature

系の初期温度を設定します。（デフォルト値：なし）

使用法：temperature = *設定温度*

使用例：temperature = 300

このコマンドは、計算を始める際の各原子の初速度を定義するために使用されます。したがって、シミュレーションが終了した後の.crd ファイルを用いてほかのインプットファイルで計算を続ける場合、このコマンドを設定する必要はありません。

solute_molecule

系の溶質分子の数を指定します。（デフォルト値：なし）

使用法：solute_molecule = *溶質分子の数*

使用例：solute_molecule = 3

5.2.4 [restraint]

原子に様々な手法で拘束をかけるための設定を行います。

method

原子に拘束をかける手法を指定します。以下のキーワードにより設定される拘束の手法が指定することができます。

position_harmonic：指定した原子グループを、指定した座標の位置にばねによる拘束を行います。

rmsd：指定した原子グループについて、指定した座標との間の rmsd(root mean square displacement)の値を満たすように拘束を行います。

以下に、method で指定した、拘束を用いる際の設定を示します。

5.2.4.1 position_harmonic

method コマンドで、position_harmonic を指定した場合、以下のコマンドを設定します。

(1)参照する座標ファイルを指定

以下のコマンドで、position_harmonic を指定する際に参照する座標ファイルを指定します。crd ファイル、pdb ファイルのどちらかで指定することができますが、指定するファイルのタイプでそれぞれ、crd_file, pdb_file を用います。

crd_file

拘束を掛ける原子を選ぶ際に参照する crd ファイルを指定します。

使用例： crd_file = test.crd

pdb_file

拘束を掛ける原子を選ぶ際に参照する pdb ファイルを指定します。

使用例： pdb_file = test.pdb

(2)拘束をかける原子グループを指定

(1)で crd ファイル、pdb ファイルのどちらで参照座標を指定したかで、それぞれ group, group_pdb を用います

group

crd_file で読み込んだ crd 構造の拘束をかける原子グループを指定します。

使用法：group = atom *選択する原子の種類　初めの残基　　終わりの残基*

使用例：group = atom non_hydrogen 24B 601B

pdb_group

pdb_file で読み込んだ pdb 構造の拘束をかける原子グループを指定します。

使用法：pdb_group = atom *選択する原子の種類　初めの残基　　終わりの残基*

使用例：pdb_group = atom non_hydrogen 24B 601B

(3)拘束条件の設定

k

拘束をかける各原子に適用するポテンシャル関数のバネ定数を指定します。

使用法： k = *ばね定数（kcal/mol/ang^2）*

使用例： k = 1.0

gradual_change_k

指定したステップ数で、k コマンドで指定したばね定数から目標のバネ定数へ徐々に変化させるように指定できます。(シミュレーション中に拘束を徐々にかけたり、またはずしたりする際に使用します。)

使用法： gradual_change_k = *ステップ数　目標のバネ定数*

使用例： gradual_change_k = 50000　0

5.2.4.2 rmsd

method で method=rmsd と設定した場合、以下のコマンドを設定します。

(1)参照する座標ファイルを指定（5.2.4.1 position_harmonic 参照）

(2)拘束をかける原子グループを指定（5.2.4.1 position_harmonic 参照）

(3)拘束条件の設定

rmsd

このコマンドで指定した rmsd 値を満たすように拘束をかけます。

使用法： rmsd = *目標とする rmsd 値（Å）*

使用例： rmsd = 3.5

gradual_change_rmsd

指定したステップ数で拘束に用いる rmsd 値を、rmsd コマンドで指定した値から目標の値へ変化させるように指定します。(Targeted MD を行う際に用います。3.3.2.2 参照)

使用法：gradual_change_rmsd = *ステップ数　目標の rmsd 値*

使用例：gradual_change_rmsd = 500000　0

best_fit

シミュレーションの各ステップで、拘束条件の rmsd をチェックする際に、シミュレーション中のスナップショットの構造を目標の座標とフィッティングする場合に指定します。

使用法： best_fit = on

k

拘束をかける各原子に適用するポテンシャル関数のバネ定数を指定します。

使用法： k = *ばね定数（kcal/mol/ang^2）*

使用例： k = 1.0

5.2.5 [constraint]

水素原子が結合している原子グループ（メチル基など）を剛体として取り扱うための設定を行います。このオプションを用いることで、水素原子の結合の振動運動を無視することができるため、シミュレーションの time step を 2 fs 程度にすることが可能となります。

rigid_body

剛体としての扱いをする対象の原子を指定します（デフォルト：なし）

使用例： rigid_body = hydrogen

5.2.6 [PT_control]

系のアンサンブルおよび、温度や圧力のコントロールを設定します。

ensemble

系の計算を行うためのアンサンブルを指定します。（デフォルト：NVT)

使用法：ensemble = {NVT|NVE|NPT|NPT_xyz|NPAT|NPT_isoxy|NPT_flexible}

使用例：ensemble = NPT

*上述の 3 種類の NPT は以下のような違いがあります。

NPT:セルの xyz 各成分均等に変化

NPT_xyz:セルの xyz 各成分独立に変化

NPAT:セルの z 成分だけ変化（xy 平面を固定）

NPT_isoxy:セルの xy 成分均等に、z 成分が独立に変化

NPT_flexible:セルの各ベクトルが変化（ボックスの頂点が直角を維持しない）

initialize

系のアンサンブルを変更する際に用います（デフォルト：なし）

使用法：initialize=on

注意

このコマンドは、これまで行ったシミュレーションを引き継いで、計算を始める際に、アンサンブルを以前の場合と変更する際に指定します。（たとえば、NVT で計算していたものを、NPT に変更する場合など）**このコマンドを用いると、以前の計算で得られた、.crd ファイルに含まれている情報を引き継がないで、リセットされてしまします。したがって、とくにアンサンブルを以前の設定から変更する必要が無いときにはこのコマンドを使用しないでください。**

method

温度と圧力をコントロールする方法を指定します。(デフォルト : extended_system)

使用法：method = {extended_system|rescaling}

使用例：method = rescaling

注意

rescaling は NVT アンサンブルのみで利用可能です。

temperature

系の温度を設定します。 （デフォルト：298.15 (K)）

使用法：temperature = *設定温度(K)*

使用例：temperature = 310

pressure

系の圧力を設定します。（デフォルト：1 (atm))

使用法：pressure = *設定圧力(atm)*

使用例：pressure = 1.0

gradual_change_T

温度を目標温度まで指定したステップ数で変化させる際に用います。（デフォルト：ステップ数=0, 目標温度=temperature と同じ）

使用法：gradual_change_T = *シミュレーションのステップ数 目標温度(K)*

使用例：gradual_change_T = 10000 300　（10000 ステップかけて温度を 300(K)にする）

5.2.7 [nonbond]

このセクションでは、系の非結合相互作用の計算に必要な設定を行います。

cutoff

非結合相互作用のカットオフ半径を指定します（デフォルト値：9.0Å）

使用法：cutoff = *カットオフ半径（Å）*

使用例：cutoff = 10

n_cell　([ewald]のセクションで d_grid を指定してある場合には必要ない)

並列計算を行う際に、ボックスをセルにより空間分割する方法を指定する。

使用法：n_cell = *(n_cell)x (n_cell)y (n_cell)z*

(n_cell)x,y,z：各成分方向のセルの数（整数）

使用例：n_cell = 20 20 20

注意

セルの指定方法は以下のルールにしたがう必要があります。

セルの最小幅は、(cutoff + 4.5)/2 で求まる値を用います。（例：cutoff = 9(Å)ならば、6.75(Å)）したがって、box の幅が 64Å、cutoff が 9 の時には、64/6.75 = 9.481 で 9 以下にします。

n_pe　([ewald]のセクションで d_grid を指定してある場合には必要ない)

並列計算に用いるプロセスを空間に分割する方法を指定します。

使用法：n_pe = *(n_pe)x (n_pe)y (n_pe)z*

(n_pe)x,y,z：各成分方向のプロセスの数（整数）

使用例：n_pe = 10 10 10

注意

プロセスの指定方法は、以下のルールを満たす必要があります。

(n_cell)x, (n_cell)y, (n_cell)z はそれぞれ(n_pe)x, (n_pe)y, (n_pe)z で割りきれる必要があります。

[ewld]のセクションで指定する x, y, z 各成分のグリッド数がそれぞれ(n_pe)x, (n_pe)y, (n_pe)z で割りきれる必要があります。

この計算に用いる全プロセス数が x, y, z 各成分のプロセス数の積、つまり (n_pe)x*(n_pe)y*(n_pe)z になるようにします。

5.2.8 [ewald]

このセクションでは、系の長距離静電相互作用を計算する PME 法(Particle Mesh Ewald)の設定を行います。

grid

系のボックスに対して定義するグリッドの XYZ 方向への数を指定します。（デフォルト値：なし）

使用法：grid = *(n_grid)x　(n_grid)y　(n_grid)z*

(n_grid)xyz はボックスの XYZ 方向へのグリッドの数（整数）

使用例：grid = 20 20 20

注意

グリッドを指定する際は、間隔が約 1.1Å 程度以下となるようにする。

d_grid

系のボックスに対してグリッドを定義する際に用いるグリッドの間隔を指定します。（デフォルト値：0）

使用法：d_grid = *グリッドの間隔　(Å)*

（ここで、グリッドの間隔は 1.0Å 程度以下とする）

使用例：d_grid = 1.1

注意

d_grid を用いるときは grid を定義は必要ありません。d_grid を決めることにより、MARBLE が、系のボックスサイズから自動的に XYZ 方向のグリッドの数を決定します。また、d_grid を用いることで、[nonbond]で定義していた、n_cell および n_pe も MARBLE が自動的に決定します。[nonbond]のセクションに示したような条件を満たす、n_pe および n_cell を見つけるのは比較的骨の折れる作業です。したがって、通常の MD 計算を行う際には、 d_grid を用いることをおすすめします。

5.2.9 [min]

このセクションでは、エネルギー最小化の計算の設定を行います。

step

エネルギー最小化計算を最急降下法で行うステップ数を指定します。(デフォルト値：0)

使用法：step = *ステップ数*

使用例：step = 1000

print_step

エネルギー最小化計算のエネルギーをアウトプットファイルに出力する頻度をステップ数で指定します。（デフォルト値：1)

使用法：print_step = *ステップ数*

使用例：print_step = 10

cg_step

エネルギー最小化計算を共役勾配法で行うステップ数を指定します。（デフォルト値：0)

使用法：cg_step = *ステップ数*

使用例：cg_step = 1000

注意

step コマンドと cg_step をあわせて指定すると、最初に step コマンドで指定したステップ数の最急降下法を実行した後に、cg_step で指定したステップ数の共役勾配法を実行します。

grad

エネルギー最小化計算の収束を判定する勾配の値を指定します。(デフォルト値：1.0e-4)

使用法：grad = 勾配の値

使用例：grad = 2.0e-4

5.2.10[md]

このセクションでは、分子動力学シミュレーションの設定を行います。

time_step

分子動力学シミュレーションで用いる 1 ステップあたりの時間刻みを指定します。（デフォルト値：0.5(fs))

使用法：time_step = *1 ステップあたりの時間刻み(fs)*

使用例：time_step = 2

step

分子動力学シミュレーションを行うステップ数を指定します。（デフォルト値：0)

使用法：step = *ステップ数*

使用例：step = 500000

prop_file

分子動力学シミュレーションのデータ出力ファイル（prop ファイル）を指定します。(デフォルト値：なし)

使用法：prop_file = *ファイル名*

使用例：prop_file = md1.prop

prop_step

prop ファイルへの出力の頻度を指定します。（デフォルト値：100）

使用法：prop_step = *ステップ数*

使用例：prop_step = 1000

trj_file

分子動力学シミュレーションのトラジェクトリを出力するファイル(trj ファイル）を指定します。（デフォルト値：なし）

使用法：trj_file = *ファイル名*

使用例：trj_file = md1.trj

trj_step

trj ファイルへの出力の頻度を指定します。（デフォルト値 ：100)

使用法：trj_step = *ステップ数*

使用例：trj_step = 1000

trj_output

trj ファイルに出力する情報を指定します。（デフォルト値：XB)

使用法：trj_output = {X|V|B}(X:座標, V:速度, B:ボックスの情報）

使用例：trj_output = XB

trj_format

trj ファイルのフォーマットを指定します。（デフォルト値：text）

使用法：trj_format = {text|dcd}

(text: テキストフォーマット、dcd: dcd フォーマット)

使用例：trj_format = dcd

trj_wrap_molecules

系の分子を周期境界の箱の中に収めるように指定します。(デフォルト値：off)

使用法：trj_wrap_molecules = {on}

使用例：trj_wrap_molecules = on

trj_xst_output

NAMD で用いられる xst ファイルを出力するように指定します。xst ファイルは、エネルギー表示法による計算を行う際に必要です。（デフォルト値：off）

使用法：trj_xst_output = {on}

使用例：trj_xst_output = on

print_step

分子動力学シミュレーションのエネルギーをアウトプットファイルに出力する頻度をステップ数で指定します。（デフォルト値：1)

使用法：print_step = *ステップ数*

使用例：print_step = 10

remove_momentum

分子動力学シミュレーション中に計算対象の蛋白質の並進や回転の運動を取り除くかどうかを指定します。（デフォルト値：off）

使用法：remove_momentun =
{off | all | all_rotation | solute_translation | solute_rotation}

使用例：remove_momentum = solute_rotation