# MODYLAS Tutorial

Version 1.0.2

4 Nov. 2014

CONTENTS

## 1. ディレクトリとファイルの概要

ダウンロードした MODYLAS のファイル MODYLAS1.0.0.tar.gz を解凍

```
%> tar zxvf MODYLAS0.1.0.0.tar.gz
```

すると、次節に示すディレクトリとファイルが生成される。

### 1.1 ディレクトリ構成

(1) MODYLAS_1.0.0/ 以下：

- binary/ MODYLAS 本体の実行モジュールを収納
- document/ ドキュメントを収納
- sample/ 入出力ファイルのサンプルを収納
- src/ MODYLAS のプログラムソースファイルを収納
- LISENCE.pdf ライセンスを記述したファイル

(2) binary/ 以下

- K/ modylas 京コンピュータで動作する MODYLAS 実行モジュール

(3) document/ 以下

- manual/ マニュアルを収納
  ReferenceManual_1.0.0.pdf … リファレンスマニュアル
- tutorial/ チュートリアルを収納
  Tutarial_1.0.0.pdf … チュートリアル

(4) sample/ 以下

- water/ 水（数万原子系）の入力ファイル
- pyp111/ タンパク質系(数万原子系)の入力ファイル
- pyp888/ タンパク質系（1000 万原子系）の入力ファイル

### 1.2 実行ファイルの種類

- modylas MODYLAS 本体
- modylas-text2bin アスキー形式からバイナリ形式への入力ファイルの変換プログラム
- modylas-mdtrj2xyz バイナリ形式での軌跡の出力ファイル sessionname.mdtrj.bin から xyz 形式への変換プログラム

（これらのパスは前節参照）

### 1.3 入出力ファイルの種類

MODYLAS の入出力ファイル名は、拡張子以外の部分はすべて共通でなければならない。以下この部分を”sessionname”と表記する。sessionname は任意の文字列でかまわないが、一文字目には数字を使用することはできない。各入出力ファイルは拡張子により区別し、その一覧を次の節で示す。

【入力ファイル】

計算条件を sessionname.mddef に記述する。力場と初期座標・速度の入力ファイルには、アスキー形式とバイナリ形式のどちらかを使うことができる。通常はアスキー形式の入力ファイルを推奨するが、巨大系の計算には、入力ファイルの読み込み時間を短縮するためにバイナリ形式の入力ファイルを使用することを推奨する。アスキーとバイナリ形式の両方が存在する場合には、バイナリ形式が優先される。

- sessionname.mddef
  計算条件（アスキー形式）
- sessionname.mdff
  力場情報（アスキー形式）
- sessionname.mdff.bin
  sessionname.mdff のバイナリ形式版（バイナリ形式）
- sessionname.mdxyz
  構成原子の座標と速度および周期セル情報（アスキー形式）
- sessionname.mdxyz.bin
  sessionname.mdxyz のバイナリ形式版（バイナリ形式）
- sessionname.posiconst　（オプション）
  原子の位置拘束(position constrain)の外部入力ファイルによる指定（アスキー形式）。原子拘束は力場入力ファイル(sessionname.mdff)で指定できるが、この sessionname.posiconst が存在するときはこちらを優先し、力場ファイル内の指定は無効になる。巨大系の場合には読み込み速度の都合上、こちらの外部入力ファイルを用いることを推奨する。バイナリ形式の力場の入力ファイル(sessionname.mdff.bin)では原子の位置座標の拘束キーワード(<position constrain>)は無効であるため、こちらの外部入力ファイルを使わなければならない。

【出力ファイル】

- sessionname.mdrun
  計算に要した経過時間情報（アスキー形式）
- sessionname.mdmntr
  時間ステップごとの物理量(ハミルトニアン,温度,圧力など)（アスキー形式）
- sessionname.mdtrj.bin
  原子座標と,速度の軌跡（バイナリ形式）
- sessionname.restart.bin
  リスタート用の座標・速度情報（バイナリ形式）。sessionname.mdxyz.bin の形式と同じ
- sessionname.restart. asc
  計算の最終構造の座標と速度が、リスタート用の入力ファイル sessionname.mdxyz の形式で出力されたもの（アスキー形式）。デフォルトで出力される設定だが、sessionname.mddef において〈output〉タグ内の変数 ascii=no と指定した場合には出力されない。

## 2. インストール

### 2.1 マシン環境

- 下記の環境で動作確認を行っています。
  京コンピューター, FX10, PC クラスター
- コンパイラ
  MPI でコンパイルするためのコマンドはシステムごとに異なりますので、システムで用意されているコマンドを確認してください。
  - [富士通] frtpx (mpifrtpx / mpifrt 等)
  - [インテル] ifort (mpif90/mpiifort 等)
  - [PGI] pgf90 (mpif90 等)
- MPI ライブラリ
  通常は、MPI 用のコンパイルコマンドでリンクすれば自動で最適な MPI ライブラリをリンクします。

### 2.2 コンパイル方法

```
tar xvzf modylas-1.0.0.tar.gz
cd modylas-1.0.0/source/
./configure --with-kind-fortran-compiler=XXX
```

*XXX* にはあらかじめ用意されたシステムのものを指定します。一致するものがない場合は最も近そうなシステムを指定します。あらかじめ用意されているシステムは、

| | |
|---|---|
| K | 京コンピューター用 |
| FX10 | 富士通 FX10 用 |
| INTEL | インテルコンパイラーを使用するシステム用 |
| PGI | PGI コンパイラーを使用するシステム用 |

です。その後、必要に応じて上記の環境変数を設定して再度 configure を行うか src/Makefile を直接編集します。

```
cd src
make
```

(注 1) configure コマンドで設定された FC の値が正しくない場合は以下の設定を行ってからもう一度 configure をやり直してください。

```
setenv FC MPIF90CMD
```

*MPIF90CMD* には MPI 用 Fortran90 コンパイラーを指定します。

(注 2) configure コマンドで設定された FCFLAGS の値が正しくない場合は以下の設定を行ってからもう一度 configure をやり直してください。

```
setenv FCFLAGS FLAGS
```

*FLAGS* には最適化のレベルを指定します。

コンパイル方法の詳細は Reference Manual を参照してください。

# 3.　サンプル１：水

この章では、簡単な例である水のシミュレーションを通じて MODYLAS の一連の計算手順を示す。水分子の熱平衡状態の軌跡を得るために、標準的な手順に従い、エネルギー最小化、NVT アンサンブルによる常温での計算、NPT アンサンブルによる常温常圧での計算を順次行う。

## 3.1 入力ファイル概要

水 4428 個の系の入力ファイルをサンプルとして sample/water/以下に置いてある。これらの初期構造および力場は、入力支援ソフトウェア、Nano-Ignition あるいは VMD ソフトウェアからの MODYLAS 用出力 plugin(公開予定)を用いることにより作成できる。ここでは入力ファイルが準備できた段階からの手順例を示す。入力ファイルの中に記述されているタグやキーワードの説明は Reference Manual に記載されている。

## 3.2　計算の実行

まず､利便性のため MODYLAS の実行ファイルを入力ファイルのあるディレクトリにコピーあるいはリンクを張っておくと作業しやすい：

```
%〉 cd  sample/water/
%〉 ln  -s  (MODYLAS のディレクトリ)/modylas  ./
```

### 3.2.1　エネルギー最小化

sessionname が water_opt のファイルがエネルギー最小化のための入力ファイルである。プログラムの実行は、使用する計算機のシステムに対応した方法に従って行う。

(1)バッチジョブを使わないシステムでは、

```
%〉 mpiexec -n 8 ./modylas  ./water_opt
```

(2)バッチジョブを使うシステムではジョブ投入スクリプトを指定のコマンドで投入する。例えば京コンピュータでは、ジョブ投入スクリプト”run_K.sh”の中の並列ノード数やファイル名を確認(Reference Manual 6.1 節のジョブ投入スクリプト記入例の①～⑧を参照)し、

```
%〉 pjsub  ./run_K.sh
```

のコマンドでジョブを投入する。

```
%〉 pjstat
```

によりジョブの状態の確認を、

```
%〉 pjdel  (job ID 番号)
```

によりジョブを削除できる。

### 3.2.2　NVT

3.2.1 のエネルギー最小化の計算結果を用いて次に NVT アンサンブルの計算を行う。次のコマンドで session name が water_nvt の 3 つの入力ファイルを生成する：

```
%〉 cp  water_opt.mddef  water_nvt.mddef
```

```
%> cp water_opt.mdff water_nvt.mdff
%> cp water_opt.restart.asc water_nvt.mdxyz
```

力場と初期構造の入力ファイルは修正する必要はないが、計算条件を変更するため water_nvt.mddef を vi や emacs などのテキストエディタで修正する必要がある：

```
%> vi water_nvt.mddef
```

あるいは

```
%> emacs water_nvt.mddef
```

修正場所は、アンサンブル、設定温度、熱浴、初速度、ステップ数などの指定である(詳細略)。参考のため、同じディレクトリに water.mddef_for_nvt という名前で修正後の入力ファイルを置いてある。これを参考にして water_nvt.mddef を修正する。

(1)バッチジョブを使わないシステムでは、

```
%> mpiexec -n 8 ./modylas ./water_nvt
```

により計算を実行する。

(2) 京コンピュータでは、ジョブスクリプト run_K.sh 内のファイル名を修正し(“water_opt” ⇒”water_nvt”), 次のコマンドで MODYLAS のジョブを投入する：

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

### 3.2.3 NPT

3.2.2 の NVT の計算結果を用いて次に NPT アンサンブルの計算を行う。前節と同様に、次のコマンドで session name が water_npt の 3 つの入力ファイルを生成する：

```
%> cp water_nvt.mddef water_npt.mddef
%> cp water_nvt.mdff water_npt.mdff
%> cp water_nvt.restart.asc water_npt.mdxyz
```

計算条件を変更するため water_npt.mddef をテキストエディタで修正する：

```
%> vi water_npt.mddef
```

あるいは

```
%> emacs water_npt.mddef
```

修正場所は、アンサンブル、設定圧力、圧力浴、の指定である(詳細略)。参考のため、同じディレクトリに water.mddef_for_npt という名前で修正後の入力ファイルを置いてある。これを参考にして water_npt.mddef を修正する。

(1)バッチジョブを使わないシステムでは、

```
%> mpiexec -n 8 ./modylas ./water_npt
```

により計算を実行する。

(2) 京コンピュータでは、ジョブスクリプト run_K.sh 内のファイル名を修正し(“water_nvt” ⇒”water_npt”), 次のコマンドで MODYLAS のジョブを投入する：

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

## 3.3 計算結果のモニター

3.2 の計算により出力されたデータの確認方法を簡単に示す。

### 3.3.1 ハミルトニアン、温度、圧力等のモニター

ハミルトニアン、温度、圧力などの物理量は、sessionname.mdmntr に sessionname.mddef 内の〈monitor〉タグ内の変数 interval で指定されたステップ間隔で逐次書き出される。sessionname.mdmntr 内に出力されている物理量および単位がヘッダーに書かれている。また Reference Manual にも掲載されている。vi や emacs などのテキストエディタあるいは less などのコマンドにより sessionname.mdmntr の中の出力値を確認することができる。グラフ表示したい場合には、sessionname.mdmntr をそのまま用いることができる。例えば gnuplot により簡単に表示可能である。gnuplot がインストールされている環境で

%〉 gnuplot

と入力し起動する。「gnuplot〉　」につづく空白に決まった文字列を打つことでグラフが画面に表示される。たとえばエネルギー最小化でのポテンシャルエネルギーを確認したい場合は、

gnuplot〉 plot ‘water_opt.mdmntr’ using 4 with lines

と打つ。ここで ‘…’ 内はファイル名を，using に続く数字の 4 は water_opt.mdmntr 内の 4 列のデータ(ポテンシャルエネルギー)を表示することを，with lines はデータ間を線でむすぶことを意味する。NPT の計算の温度の時間依存性を見たいときには、

gnuplot〉 plot ‘water_npt.mdmntr’ using 2:7 with lines

と打つ。ここで using に続く 2:7 は 2 列目のデータに対して 7 列目のデータを表示することを意味する。

### 3.3.2 軌跡のアニメーション表示

MODYLAS の計算により指定の時間ステップ間隔で軌跡（各原子の座標と速度）が.mdtrj.bin の拡張子のファイルに出力される。例として NPT の計算で得た water_npt.mdtrj.bin から動画表示の手順を示す。

water_npt.mdtrj.bin はバイナリ形式である。そのためまずは座標についてアスキー形式に変換する。これには付属の変換プログラム： modylas-mdtrj2xyz を用いることにより、xyz 形式の軌跡ファイルを生成することができる。入出力ファイルの存在するディレクトリに入り、変換プログラムの実行モジュール modylas-mdtrj2xyz にリンクを張る：

%> ln –s (ディレクトリのパス)/modylas-mdtrj2xyz ./

次に以下のコマンドを実行する：

%> ./modylas-mdtrj2xyz ./water_npt

この時、出力させたい原子数を記入し enter を押す。全原子数を指定したい場合には、マイナスの数字、例えば「-1」とすればよい。次に何コマごとに出力するかを尋ねられる。出力データの全コマを変換したい時には「1」を入力し enter を押す。すると、water_npt.xyz というファイルが生成される。動画表示した場合には、別途動画表示ソフトウェアを用いる。ここでは代表的なソフトウ

ェアの VMD（別途インストールが必要）を例とする。

```
%>  vmd   ./water_npt.xyz
```

により動画を表示できる。

## 4. サンプル 2：水溶液中のタンパク質（1000 万原子系）

この章では、京コンピュータを用いた水溶液中のタンパク質をサンプルとして超巨大系(1000 万原子系)の計算手順を示す。

### 4.1 インプットファイル概要

1000 万原子系のような巨大系を扱う時に、サンプル 1 の時のようなアスキー形式の入力ファイルを用いると、読み込みだけで長い時間を費やしてしまうため、バイナリ形式に変換した入力ファイルを用いる。sample/pyp888/以下に入力ファイル一式が置かれている。

### 4.2 計算の実行

まず､利便性のため MODYLAS の実行ファイルを入力ファイルのあるディレクトリにコピーあるいはリンクを張っておくと作業しやすい：

```
%〉 cd  sample/pyp888/
%〉 ln  -s  (MODYLAS のディレクトリ)/modylas  ./
```

sessionname が pyp888 のファイルが入力ファイルである。プログラムの実行は、ジョブ投入スクリプト”run_K.sh”の中の並列ノード数やファイル名を確認(64 ノードに設定してある。Reference Manual 6.1 節のジョブ投入スクリプト記入例の①～⑧を参照)し、

```
%〉 pjsub  ./run_K.sh
```

のコマンドでジョブを投入する。

```
%〉 pjstat
```

によりジョブの状態の確認を、

```
%〉 pjdel  (job ID 番号)
```

によりジョブを削除できる。

### 4.3 計算結果のモニター

計算結果のモニターおよび軌跡の可視化は、前章の水のサンプルの場合と同じであるのでそちらを参照。

## ［付録］ 京コンピュータ 65,536 ノードを使った計算

ここでは MODYLAS と京コンピュータ 65,536 ノードを使った計算のセットアップ方法について解説する.

(1) 系の設計

Reference Manual 8.2 節で述べたように MODYLAS では各プロセスに領域分割されたセルブロックが割り当てられる. セルブロックには 1 つ以上のサブセル含まれる必要があること, かつサブセル総数が $8^n$ である制限から, 65,536 ノードを使用する場合には $n = 6$ 以上でなければならない ($8^5 < 65{,}536 < 8^6$). 一方, サブセルの一辺長さには Lennard-Jones カットオフ半径の 1/2 以上でなければならないとの制約がある. したがって基本セルの一辺長さは cutoff/2 * $2^6$ 以上にとる必要がある. 実用上, 等温等圧 *NPT* アンサンブルにおいては時間とともに基本セル一辺長さの揺らぎが生じるため, ある程度の余裕をもって基本セル一辺長さを決めておくとよい.

(2) mddef の記述例

通常, 時間刻み dt などの計算条件については並列数によらない. したがってここでは 65,536 ノードでの計算を行なう際に書き換えるべき箇所についてのみ解説する.

```
<mpi>
 division=manual   # manual/auto
 nxdiv=64          # omitted, when division=auto
 nydiv=32          # omitted, when division=auto
 nzdiv=32          # omitted, when division=auto
</mpi>
```

まずプロセス分割の方法を自動（division=auto）から手動（division=manual）に切り替える. その上で x, y, z 方向へのプロセス分割数 nxdiv, nydiv, nzdiv を 64, 32, 32 に指定する（64x32 x32=65,536).

```
type=fmm
<fmm>      nmax=4   ULswitch=3   ncell=64      </fmm>
```

静電相互作用の計算方法 type を fmm に指定する． 各軸方向のセル分割数 ncell=64 に指定する. さらに通信の最適化のため ULswitch=3 に指定する.

(3) 実行スクリプトの記述例

以下に, 京コンピュータの言語環境 K-1.2.0-14 を前提に 65,536 ノードを使った 65,536 プロセス並列計算を流す際の実行スクリプト例を示す.

```
#!/bin/sh
```

```
#PJM --rsc-list "node=48x54x32:strict"
#PJM --rsc-list "elapse=1:00:00"
#PJM --rsc-list="rscgrp=huge"
#PJM --stg-transfiles all
#PJM --mpi "use-rankdir"
#PJM --mpi "rank-map-hostfile=./hostfile-48x54x32-64x32x32-all"
#PJM --stgin    "rank=* ./modylas     %r:./"
#PJM --stgin    "rank=* ./pyp888.mddef    %r:./"
#PJM --stgin    "rank=* ./pyp888.mdff.bin %r:./"
#PJM --stgin    "rank=* ./pyp888.mdxyz.bin %r:./"
#PJM --stgin    "rank=* ./hostfile-48x54x32-64x32x32-all %r:./"
##
#PJM --stgout "rank=0 %r:./pyp888.mdmntr ./ "
#PJM --stgout "rank=0 %r:./pyp888.mdrun ./ "
#PJM --stgout "rank=0 %r:./pyp888.restart.bin ./ "
#PJM --stgout "rank=0 %r:./pyp888.mdtrj.bin ./    "
#PJM -s

# path
. /work/system/Env_base_1.2.0-15

# Parameters
NPROCS=65536
NTHREADS=8
export PARALLEL=${NTHREADS}
export OMP_NUM_THREADS=${NTHREADS}
LPG="/opt/FJSVxosmmm/sbin/lpgparm -t 4MB -s 4MB -h 4MB -d 4MB -p 4MB"

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

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

LD="./modylas ./pyp888"
mpiexec -n ${NPROCS} ${MCA_PARAM} ${LPG} ${LD}
```

京コンピュータ 65,536 ノードを使った計算を MODYLAS の最大効率をもって流すためには, 2 つの特別な操作を行なう必要がある.

(1) 論理ノード形状の変更・・・京コンピュータの論理ノード形状は標準で 48x54x32 に設定されている. これを 64x32x32 へと変更する. この変更は#PJM --mpi "rank-map-hostfile= "において指定された map-hostfile により行う. map-hostfile の書式については京コンピュータマニュアルを参照のこと.

(2) 全ノードの占有・・・map-hostfile による 64x32x32 への論理ノード形状変更には, 全ノードを占有する必要がある. #PJM --rsc-list "node=48x54x32:strict" でこれを行なう.

これら 2 つの操作に加えて, mddef に記載したプロセス形状 nxdiv,nydiv,nzdiv を変更した論理ノード形状と一致させておく.

上の実行スクリプト例ではステージインさせるインプット一式および実行バイナリ modylas を#PJM –stgin の行において指定している. 一方, ステージアウトさせるアウトプット一式を#PJM –stgout の行において指定している. プロセス数およびスレッド数については通常通り NPROCS および NTHREADS において指定している. インプットバイナリをリトルエンディアン形式で作成した場合には, エンディアン変換のための環境変数 FORT90L='-Wl,-T'を設定しなければならない. MODYLAS およびインプットを LD="./modylas ./pyp888"において指定し, mpiexec -n ${NPROCS} ${MCA_PARAM} ${LPG} ${LD} において並列実行させている.

その他の詳細については京コンピュータマニュアルを参照。