μ² lib チュートリアル

寺田　透（東京大学）
森次　圭（横浜市立大学・理研）

「だれにでもわかる拡張サンプリングシミュレーション–実習編–」
2014年7月24日（木） 於 産業技術総合研究所・ゲノム情報研究センター

構成

1. μ²libの概要
2. μ²libのコンパイル
3. Single copyシミュレーション実習
4. マルチコピーシミュレーションの概要
5. MSESの概要
6. MSESシミュレーション実習
7. ストリング法の概要
8. ストリング法実習

1. μ²libの概要

- A class library for developing multi-copy and multi-scale simulation programs
  マルチコピー・マルチスケール分子シミュレーション法開発の基盤となるクラスライブラリ
- 理化学研究所次世代計算科学研究開発プログラム分子スケールチームの寺田　透、森次圭、松永康佑、木寺詔紀により開発
- GNU General Public Licenseの下で無償公開
  ダウンロードサイト：http://www.mu2lib.org/

開発の経緯

- 生物学上重要な現象（複合体形成、立体構造変化など）の時間スケールはミリ秒以上
  - 通常の分子動力学法では追跡できない

- 統計力学に基づいてマルチコピー・マルチスケールアプローチが有効
- 新規マルチコピー・マルチスケール法開発を支援するライブラリを開発

μ²libの特徴

- C++で実装
  - オブジェクト指向：シミュレーション対象の系を1つのオブジェクトとして扱う
    →マルチコピー・マルチスケールへの拡張が容易
  - カプセル化：データの意図しない変更を防ぐ、アプリケーションに影響を与えずに実装変更が可能
  - 継承・多態性：容易に既存のクラスを拡張し、新規アルゴリズムを実装できる
- OpenMPおよびMPIによるハイブリッド並列化

μ²libの構成

- コアクラス群
  - 相互作用クラス
  - 入出力クラス
  - シミュレーションクラス
- インターフェイスクラス群
  - Amberフォーマットファイル入出力用の派生クラス
- アプリケーションクラス群
  - マルチコピー・マルチスケールアプリケーション用派生クラス

コアクラス群

- 相互作用クラス
  - 共有結合長、共有結合角、二面角相互作用
  - 非共有結合相互作用（カットオフ法）
  - Particle mesh Ewald法
  - Generalized Born法
  - 距離・角度・二面角束縛、位置束縛、Cap束縛
  - SHAKE、SETTLE
- シミュレーションクラス
  - エネルギー最小化
  - 定温・定圧シミュレーション（Berendsen、Langevin）

入出力ファイル

- 入力ファイル
  - 相互作用パラメータ
  - シミュレーション設定
  - 再スタート
- 出力ファイル
  - ログ
  - トラジェクトリ（座標、速度、エネルギー）
  - 再スタート
- インターフェイスクラス群の派生クラスを使用することで、Amber互換フォーマットで入出力可能

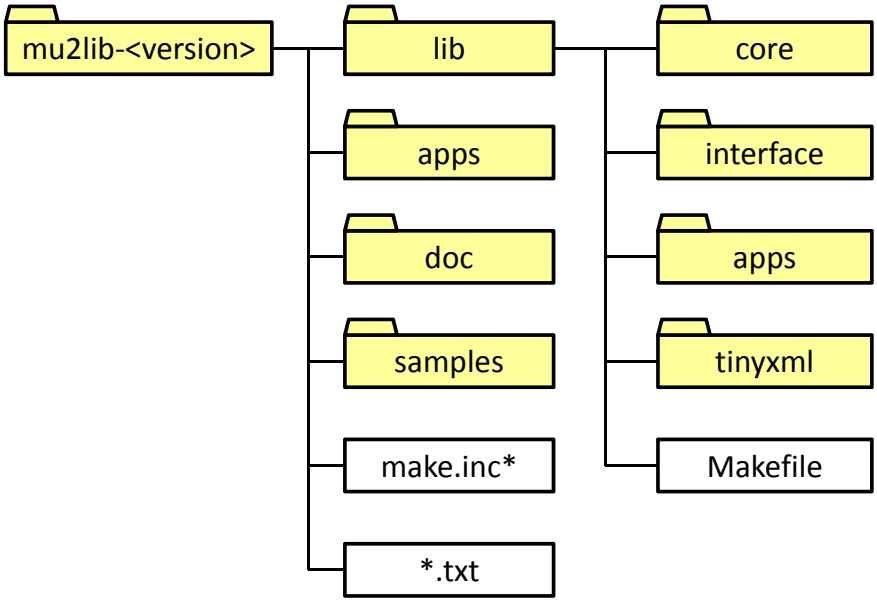
2. μ²libのコンパイル

- システム要件
  - Linuxまたは互換のオペレーティングシステム
  - GCCまたはIntel compiler、Fujitsu compiler
- 必須ライブラリ
  - GCCまたはIntel compilerの場合
    - MPIライブラリ：OpenMPIまたはIntel MPI
    - FFTライブラリ：FFTWまたはIntel math kernel library
    - 線形代数演算ライブラリ：LAPACKまたはIntel math kernel library
    - データ形式ライブラリ：netCDF（HDF5はオプション）
  - Fujitsu compilerの場合
    - FFTライブラリ：FFTWまたはFujitsu SSL II
    - 線形代数演算ライブラリ： LAPACKまたはFujitsu SSLII
    - データ形式ライブラリ：netCDF（HDF5はオプション）

コンパイルの手順

1. 必要に応じてFFTライブラリ、線形代数演算ライブラリ、データ形式ライブラリをインストール
2. ダウンロードサイトからmu2lib-<version>.tgzをダウンロードし、適当なディレクトリの下に展開
3. 生成したmu2lib-<version>ディレクトリに移動
4. 適切なmake.inc.*ファイルをmake.incにコピー
5. make.incでコンパイラのオプションやライブラリのインストールパスを適切に設定
6. libディレクトリに移動し、make

ディレクトリ構成



make.inc.*

- make.incのテンプレート
  - make.inc.k: 京コンピュータ用
  - make.inc.fx10: FX10（SCLS）用
  - make.inc.linux.gcc: Linux GCC用
  - make.inc.linux.intel: Linux Intel compiler用
- make.incの記述例

```
CC      = /usr/local/openmpi-1.4.5/bin/mpic++
F77     = /usr/local/openmpi-1.4.5/bin/mpif90
OPT     = -O3 -fopenmp
NETCDF  = /usr/local/netcdf-4.1.1
HDF5    = /usr/local/hdf5-1.8.6
以下略
```

サンプルアプリケーション

- appsディレクトリに収納
  - single_md: シングルコピーシミュレーション
  - cg: 粗視化シミュレーション
  - mses_cc: Multiscale enhanced sampling
  - replica1d_cc: レプリカ交換／ストリング法
  - calc_dist: トラジェクトリ解析プログラム
- 各アプリケーションのディレクトリに移動し、make
- samplesディレクトリに各アプリケーションの入力ファイルのサンプルを収納

single_md（1）

- main.C（一部改変）

```
#include "mu2lib.h"
#include "mu2lib_interface.h"
int main(int ac,char **av)
{
  AmberParam p;
  AmberConf c;
  AmberLog *lg=NULL;
  Molecule m;
  Dynamics a;
  Parallel para;
  int imin, ntt, nsteps;
  bool restart;

  para.init(&ac,&av);
  para.simple_mode();
```

```
  if(para.get_local_master()) {
    c.parse_args(ac,av);
    c.read();
    c.open_output_files();
    p.read(c);
    lg=(AmberLog*) c.get_logger();
    lg->print_file_assignment(c);
    lg->print_param(p);
    lg->print_conf(c);
    m.read_restart(c);
  }

  c.bcast(para);
  p.bcast(para);
  a.bcast(para);
```

single_md（2）

```
  a.setup(p,c,para,&m);

  imin=c.get_imin();
  if(imin == 0) {
    ntt=c.get_ntt();
    nsteps=c.get_nsteps();
    restart=c.get_restart();
    if(ntt == 0) {
      a.leapfrog(nsteps,restart);
    } else if(ntt == 1) {
      a.berendsen(nsteps,restart);
    } else if(ntt == 3) {
      a.langevin(nsteps,restart);
    }
  } else if(imin == 1) {
    a.minimize(c);
  }
  para.finalize();
}
```

- AmberConf、AmberParam、AmberLog、AmberRestart
  - Conf、Param、Log、Restart クラスのAmberフォーマット入出力用派生クラス
- Parallel
  - MPIによる並列計算を制御
- Molecule
  - シミュレーション対象を記述
- Dynamics
  - シミュレーションを実行

マニュアル

- doc/mu2lib.html
- http://www.mu2lib.org/documents.html
- 記載内容
  - ライブラリとアプリケーションのコンパイル方法
  - アプリケーションの実行方法
  - 各クラスのprotected fieldとpublic methodの解説

3. Single copyシミュレーション実習

1. 入力ファイルの作成
2. シミュレーションの実行
   - エネルギー最小化
   - 平衡化
   - プロダクション
3. 結果の解析

3.1. 入力ファイルの作成（1）

- 10残基のミニタンパク質chignolinの水溶液中におけるシミュレーションを行う
  - ChignolinのPDB ID: 1UAO
- AmberToolsのLEaPを用いてパラメータファイルを作成する
  - AmberTools: http://ambermd.org/#AmberTools
  - マニュアル：http://ambermd.org/doc12/

3.1. 入力ファイルの作成（2）

- チュートリアルファイルのコピー

```
> cp -r /home/islim/mu2lib-k/tutorial ~/
```

- 作業ディレクトリへの移動

```
> cd ~/tutorial/single_md
```

- 内容の確認

```
> ls
1uao.pdb  eq.sh    leap.sh  min.sh   prod.sh
eq.in     leap.in  min.in   prod.in  traj.in
```


3.1. 入力ファイルの作成（3）

- leap.in

```
source leaprc.ff99SBildn
x=loadPDB 1uao_01.pdb
addIons x Na+ 0
solvateBox x TIP3PBOX 9.0 iso
savePDB x leap.pdb
saveAmberParm x leap.top leap.crd
quit
```

- leap.sh

```
#!/bin/sh
perl <<END
open(IN,"1uao.pdb");
open(OUT,">1uao_01.pdb");
while(<IN>) {
  last if(/^MODEL/);
}
# Extract first model
while(<IN>) {
  last if(/^ENDMDL/);
  print OUT;
}
close(IN);
close(OUT);
END

AMBERHOME=/home/islim/mu2lib-k/amber12
export AMBERHOME
rm -f leap.log leap.crd leap.top parm.pdb
$AMBERHOME/bin/tleap -f leap.in
```

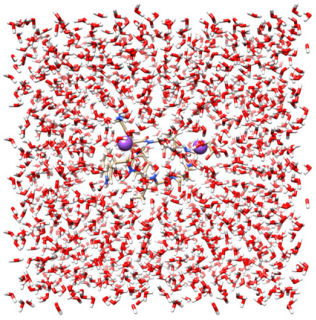
3.1. 入力ファイルの作成（4）

- leap.shの実行

```
> ./leap.sh
```

- 結果の確認

```
> ls
1uao_01.pdb  eq.sh     leap.log  leap.top  prod.in
1uao.pdb     leap.crd  leap.pdb  min.in    prod.sh
eq.in        leap.in   leap.sh   min.sh    traj.in
```



leap.pdbのイメージ

3.2. シミュレーションの実行（1）

1. エネルギー最小化
   – 使用ファイル：min.in、min.sh
2. 平衡化（100 ps束縛付き定温MD）
   – 使用ファイル： eq.in、eq.sh
3. プロダクション（100 ps定温定圧MD）
   – 使用ファイル： prod.in、prod.sh

3.2.1. エネルギー最小化

- min.in

```
Energy minimization
 &cntrl
    imin=1,
    ntx=1, irest=0, ntrx=1,
    ntxo=1, ntpr=1, ntwr=500,
    ntwx=0, ioutfm=1,
    ntr=0,
    maxcyc=500, ncyc=100, ntmin=1,
    ntc=1, tol=1.0e-6,
    ntb=1, cut=8.0, igb=0,
 /
```

- min.sh

```
#!/bin/sh
#------pjsub options ------#
#PJM -L "rscgrp=small"
#PJM -L "node=1"
#PJM --mpi "proc=1"
#PJM -L "elapse=10:00"
#PJM -j
#------Program Execution --------#
MU2LIB=/home/islim/mu2lib-k/mu2lib-
K-2.0
mpiexec ¥
$MU2LIB/apps/single_md/single_md ¥
  -O ¥
  -i min.in ¥
  -o min.out ¥
  -p leap.top ¥
  -c leap.crd ¥
  -r min.restrt
```


3.2.2.平衡化

- eq.in

```
Equilibration
 &cntrl
    imin=0,
    ntx=1, irest=0, ntrx=1,
    ntxo=1, ntpr=50, ntwr=500,
    ntwx=500, ioutfm=1,
    ntr=1, restraint_wt=10.0,
    restraintmask="@CA"
    nstlim=50000, nscm=500, dt=0.002,
    ntt=1, temp0=300.0, tempi=0.0,
    tautp=2.0,
    ntp=0, pres0=1.0, taup=2.0,
    ntc=2, tol=1.0e-6,
    ntb=1, cut=8.0, igb=0,
 /
```

- eq.sh

```
#!/bin/sh
#------pjsub options ------#
#PJM -L "rscgrp=small"
#PJM -L "node=2"
#PJM --mpi "proc=16"
#PJM -L "elapse=15:00"
#PJM -j
#------Program Execution --------#
MU2LIB=/home/islim/mu2lib-k/mu2lib-
K-2.0
mpiexec ¥
$MU2LIB/apps/single_md/single_md ¥
  -O ¥
  -i eq.in ¥
  -o eq.out ¥
  -p leap.top ¥
  -c min.restrt ¥
  -r eq.restrt ¥
  -x eq.crd
```

3.2.3. プロダクション

- prod.in

```
Production run
 &cntrl
    imin=0,
    ntx=5, irest=1, ntrx=1,
    ntxo=1, ntpr=50, ntwr=500,
    ntwx=500, ioutfm=1,
    ntr=0,
    nstlim=50000, nscm=500, dt=0.002,
    ntt=3, temp0=300.0, gamma_ln=2.0,
    ntp=1, pres0=1.0, taup=2.0,
    ntc=2, tol=1.0e-6,
    ntb=1, cut=8.0, igb=0,
 /
```

- prod.sh

```
#!/bin/sh
#------pjsub options ------#
#PJM -L "rscgrp=small"
#PJM -L "node=2"
#PJM --mpi "proc=16"
#PJM -L "elapse=15:00"
#PJM -j
#------Program Execution --------#
MU2LIB=/home/islim/mu2lib-k/mu2lib-
K-2.0
mpiexec ¥
$MU2LIB/apps/single_md/single_md ¥
  -O ¥
  -i prod.in ¥
  -o prod.out ¥
  -p leap.top ¥
  -c eq.restrt ¥
  -r prod.restrt ¥
  -x prod.crd
```


3.2. シミュレーションの実行（2）

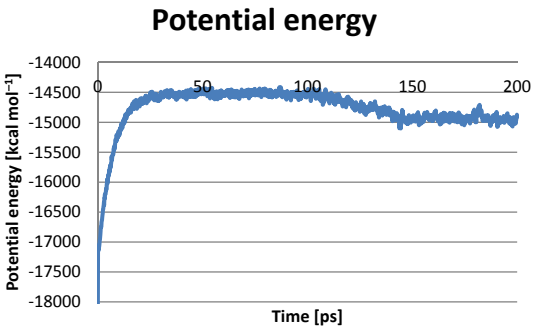
- 設定ファイル（*.in）の書式はAmberのsanderモジュールと同じ
  - マニュアル：http://ambermd.org/doc12/
- ジョブはバッチジョブとして実行
  - ジョブが終了してから次のジョブを投入すること
  - 実行状況はpjstatコマンドで確認できる

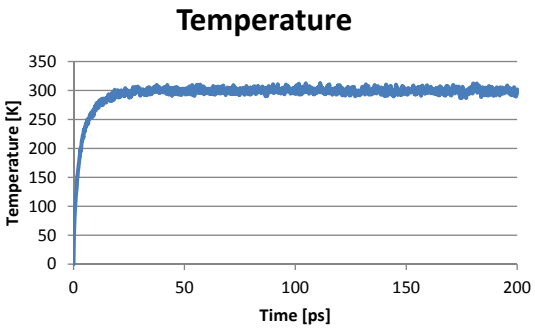
```
> pjsub min.sh
> pjsub eq.sh
> pjsub prod.sh
```

  - エネルギー最小化は約10秒、MDは約11分かかる

3. 結果の解析（1）

- ログファイル（*.out）から、エネルギー等の値の時間変化の情報を得ることができる
- トラジェクトリファイルは、UCSF chimeraやVMDなどで可視化できる
  - http://www.cgl.ucsf.edu/chimera/
  - http://www.ks.uiuc.edu/Research/vmd/
- 本実習では、水素結合距離の解析を行う

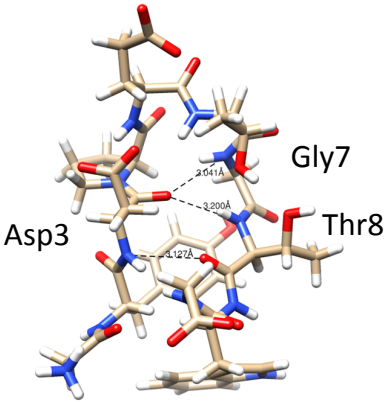




3. 結果の解析（2）

- traj.in

```
TOPOLOGY        leap.top
TRAJECTORY      prod.crd NETCDF
START           1
INTERVAL        1
DISTANCE        :3@O :7@N
DISTANCE        :3@O :8@N
DISTANCE        :3@N :8@O
RMSD            leap.pdb @CA @CA
OUT             dist.csv
```

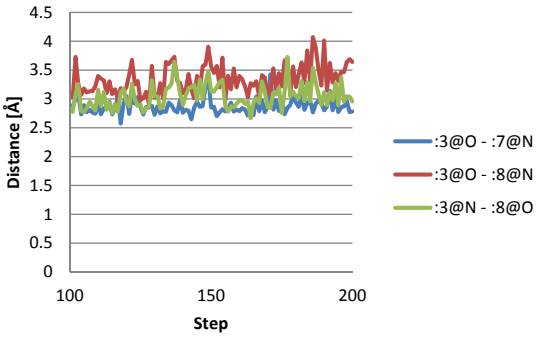


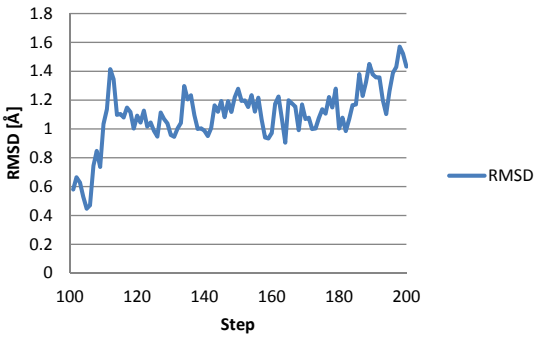
- calc_distの実行

```
> pjsub --interact -L "node=1" --mpi "proc=1"
> /home/islim/mu2lib-k/mu2lib-K-2.0/apps/calc_dist/calc_dist
> exit
```

3. 結果の解析(3)

- 結果はOUTで指定したファイルに出力される
- CSV形式で出力されるのでExcel等でそのまま開くことができる

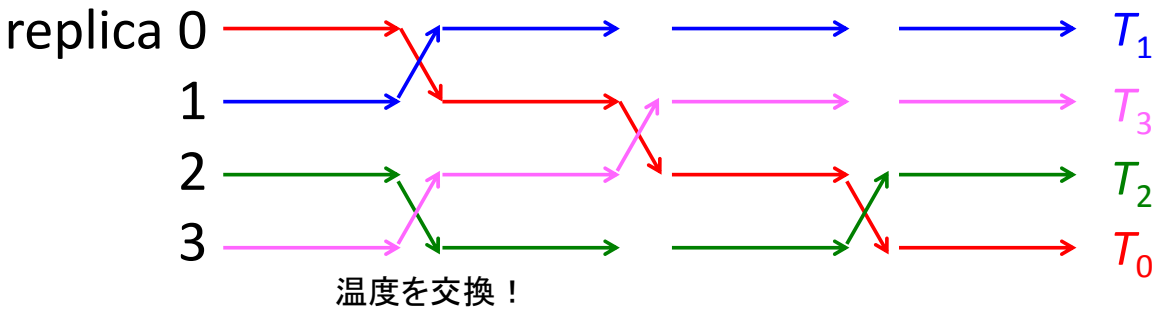




4. マルチコピーシミュレーションの概要

- 通常の平衡MD(brute-force MD)では、ミリ秒より遅い、タンパク質の機能発現に関わる運動を直接シミュレートすることは難しい
- マルチコピーを用いた統計力学的手法により、一つの準安定構造にとどまることなく効率的に構造サンプリングを行うことができる
- μ^2 libで実装済みのマルチコピーシミュレーショ
  - 温度レプリカ法
  - MSES法(本チュートリアルで実行)
  - ストリング法(本チュートリアルで実行)

温度レプリカ法



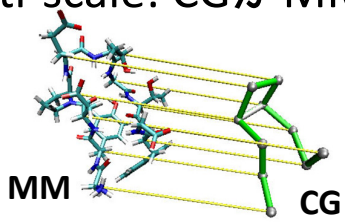
- 温度の異なるレプリカを多数並列実行
- 定期的に（各温度でカノニカル分布になるように）温度を入れ替えることで構造探索空間を広げる
- References:
  - Hansmann, *Chem Phys Lett* **281**, 140 (1997).
  - Sugita, Okamoto, *Chem Phys Lett* **314**, 141(1999).

5. MSESの概要（1）

- MSES (multiscale enhanced sampling)：粗視化(coarse-grained: CG)自由度の系をうまく用いて全原子系(MM)の構造探索を効率化するマルチスケールなシミュレーション手法。
- References:
  - Moritsugu, Terada, Kidera, *J. Chem. Phys.* **133**, 224105 (2010).
  - Moritsugu, Terada, Kidera, *J. Am. Chem. Soc.* **134**, 7094 (2012).

5. MSESの概要（2）

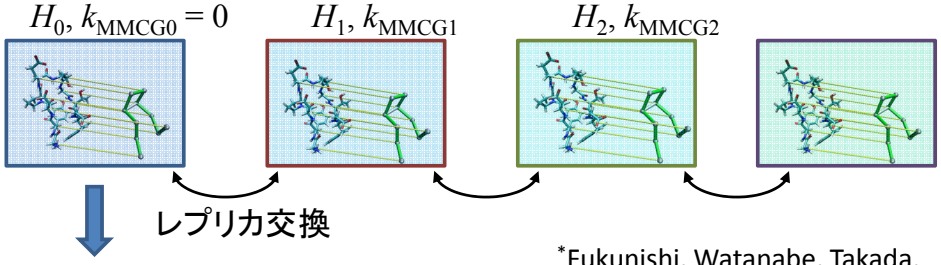
- Multi-scale: CGがMMをドライブする



$$H(\mathbf{r}_{MM},\mathbf{r}_{CG}) = H_{MM}(\mathbf{r}_{MM}) + H_{CG}(\mathbf{r}_{CG}) + k_{MMCG}V_{MMCG}(\theta[\mathbf{r}_{MM}],\mathbf{r}_{CG})$$

MM/CG constraints（k_{MMCG} V_{MMCG}）

- Multi-copy: Hamiltonianレプリカ交換法*によりMM/CGのバイアスを除く



バイアスのない構造アンサンブル!

*Fukunishi, Watanabe, Takada, *J. Chem. Phys.* **116**, 9058 (2002).

MSESの流れ

1. CGMD: CGの力場とそのパラメタを決める
2. MM/CG MD (single copy)
   - MM/CG拘束のパラメタ（$mass_{CG}$, dt_{CG}, 最大のk_{MMCG}など）を決める
3. MSES (replica)
   - レプリカ数とMM/CG拘束の配置（{k_{MMCG}}）を決めてからプロダクトラン

6. MSESの実習

1. CGシミュレーションの実行
2. MSESシミュレーションの実行
3. 結果の解析

6.1. CGMD 入力ファイル作成

- 作業ディレクトリへの移動

```
> cd ~/tutorial/mses/cg
```

md.in (Amber inputからの追加分)

```
MENM model simulation
 &cntrl
          imin=0, nmropt=0,
          ntx=1, irest=0, ntrx=1,
          ……
/
 &cgmd
           ncg=4, kcg=1.0, rcut=11.5,
           masscg=100.0,
           cg_nmix=2, cg_t=20000.0,
           cg_e=0.0, 0.0,
/
```

run.sh

```
#!/bin/sh
#------pjsub options ------#
#PJM -L "rscgrp=small"
#PJM -L "node=1"
#PJM --mpi "proc=4"
#PJM -L "elapse=10:00"
#PJM -j
#------Program Execution --------#
MU2LIB=/home/islim/mu2lib-k/mu2lib-K-2.0
mpiexec ¥
$MU2LIB/apps/cg/cg ¥
 -O ¥
 -i md.in ¥
 -o md.out ¥
 -c ake_ca.rst ¥
 -ref1 4ake_ca.crd ¥
 -ref2 1ake_ca.crd ¥
 -r md.rst ¥
 -x md.crd
```

4コアで計算

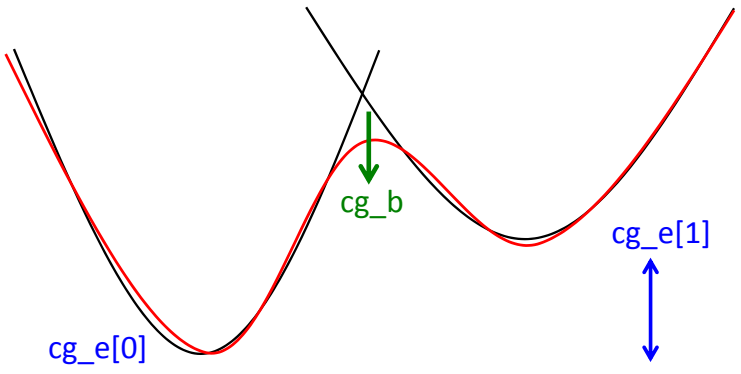
6.1. CGMDの入力パラメタ

```
ncg = 0   (not CGMD, default)
      1   (elastic network model[1]: -ref1)
      3   (plastic network model[2]: -ref1,-ref2)
      4   (mixed elastic network model[3]: -ref1,-ref2)
      5   (microscopic plastic network model[4]
           : -ref1,-ref2)

masscg: mass for each Cα atom
kcg: scaling factor for V_CG (V = kcg x V_CG)
kcons, rcut: force constant and cutoff length for ENM
cg_nmix: number of reference states (usually, = 2)
cg_t: mixing temperature
cg_e: energy of each reference state
```


[1] M. Tirion et al., Phys. Rev. Lett. 77, 1905 (1996)
[2] P. Maragakis et al., J. Mol. Biol. 352, 807 (2005)
[3] Q. Lu et al., J. Am. Chem. Soc. 130, 4774 (2008)
[4] W. Zheng et al., Proteins 69, 43 (2007)

mixed elastic network model

$$V_{CG} = -\beta^{-1} \ln\left[\exp(-\beta V_1) + \exp(-\beta V_2)\right]$$



6.1 CGMD 実行

```
$ pjsub run.sh
```

生成ファイル

```
md.out:  出力ファイル、入力パラメタの確認や各ステップのエネルギー値
md.rst:  リスタートファイル
md.crd:  座標トラジェクトリファイル
```

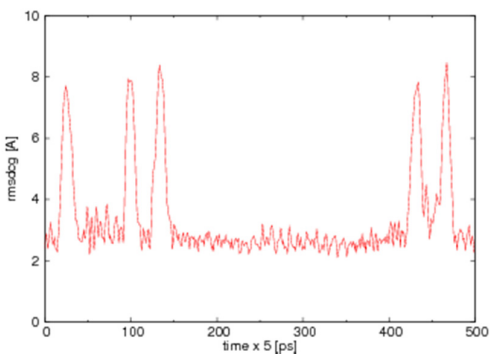

6.1 CGMD 結果

ptrajでRMSD計算（ptraj.sh）

```
#!/bin/tcsh
setenv AMBERHOME /home/islim/mu2lib-k/amber12
$AMBERHOME/bin/ptraj 214ca.top << EOF
trajin md.crd
reference ake_ca.rst
rms reference out rmsdcg.dat :1-214
go
EOF
```

実行、gnuplotで見る (またはpngファイル作成)

```
$ ./ptraj.sh

$ gnuplot
Terminal type set to 'x11'
gnuplot> plot 'rmsdcg.dat' w l
gnuplot> quit

$ gnuplot plot.gnu
$ convert rmsdcg.eps rmsdcg.png
```



6.2. MSES 入力ファイル作成

- 作業ディレクトリへの移動

```
> cd ~/tutorial/mses/msesrep
```

MM: mdgb0.in (Amber input からの追加分)

```
molecular dynamics run
 &cntrl
    imin=0, nmropt=0,
    ntx=5, irest=1, ntrx=1,
    ...
/
 &cgmd
    ncg=10,
nemmcg=0,lstmcg='listmmcg',kmmcg=0.0,
/
```

CG: mdcg.in (Amber input からの追加分)

```
molecular dynamics run
 &cntrl
    imin=0, nmropt=0,
    ntx=1, irest=0, ntrx=1,
    ...
/
 &cgmd
    ncg=14, kcg=1.0, rcut=11.5,
    masscg=100.0,
    cg_nmix=2, cg_t=20000.0,
    cg_e=0.0, 0.0,
/
```


groupfile

```
-i mdgb0.in  -o md0.out -p 1uao.top -c mdgb.restrt -r md0.restrt -x mdcrd.000
-rem 1 -remlog rem.log
-i mdgb1.in  -o md1.out -p 1uao.top -c mdgb.restrt -r md1.restrt -x mdcrd.001
-rem 1 -remlog rem.log
-i mdgb2.in  -o md2.out -p 1uao.top -c mdgb.restrt -r md2.restrt -x mdcrd.002
-rem 1 -remlog rem.log
-i mdgb3.in  -o md3.out -p 1uao.top -c mdgb.restrt -r md3.restrt -x mdcrd.003
-rem 1 -remlog rem.log
```

groupfilecg

```
-i mdcg.in  -o mdcg0.out -c 1uaoCG.crd -ref2 1uao_extCG.crd -ref1 1uaoCG.crd -
r mdcg0.restrt -x mdcgcrd.000 -rem 1 -remlog rem.log
-i mdcg.in  -o mdcg1.out -c 1uaoCG.crd -ref2 1uao_extCG.crd -ref1 1uaoCG.crd -
r mdcg1.restrt -x mdcgcrd.001 -rem 1 -remlog rem.log
-i mdcg.in  -o mdcg2.out -c 1uaoCG.crd -ref2 1uao_extCG.crd -ref1 1uaoCG.crd -
r mdcg2.restrt -x mdcgcrd.002 -rem 1 -remlog rem.log
-i mdcg.in  -o mdcg3.out -c 1uaoCG.crd -ref2 1uao_extCG.crd -ref1 1uaoCG.crd -
r mdcg3.restrt -x mdcgcrd.003 -rem 1 -remlog rem.log
```

run.sh

```
#!/bin/sh
#------pjsub options ------#
#PJM -L "rscgrp=small"
#PJM -L "node=1"
#PJM --mpi "proc=16"
#PJM -L "elapse=10:00"
#PJM -j
#------Program Execution --------#
MU2LIB=/home/islim/mu2lib-k/mu2lib-K-2.0
mpiexec ¥
$MU2LIB/apps/mses_cc/mses_cc ¥
  -O  ¥
  -i mdgb0.in ¥
  -ng 4 ¥
  -groupfile groupfile ¥
  -groupfilecg groupfilecg
```

16 コア = 4 コア × 4 レプリカで計算

6.2 MSES 実行

```
$ pjsub run.sh
```

生成ファイル

```
md[0-3].out: MM出力ファイル
md[0-3].restrt: MMリスタートファイル
mdcrd.00[0-3[]: MM座標トラジェクトリファイル
mdcg[0-3].out: CG出力ファイル
mdcg[0-3].restrt: CGリスタートファイル
mdcgcrd.00[0-3[]: CG座標トラジェクトリファイル
rem.log: レプリカ交換のログファイル
```

(rem.log: 4 replicas, # of exchange = 100)

```
#Replica setup
#Indx=    0 Replica Temp=    0.0000000 mtable=    0
#Indx=    1 Replica Temp=    0.0200000 mtable=    1
#Indx=    2 Replica Temp=    0.0500000 mtable=    2
#Indx=    3 Replica Temp=    0.1000000 mtable=    3

# Rep#, Velocity Scaling, T, EMMCG, kmmcg, Newkmmcg, Success rate (i,i+1), ResStruct#
# exchange        1
   0      1.00 350.7666162       47.54   0.0000000   0.0200000      0.00      -1
   1      1.00 350.7666162       47.54   0.0200000   0.0000000      2.00      -1
   2      1.00 350.7666162       47.54   0.0500000   0.1000000      0.00      -1
   3      1.00 350.7666162       47.54   0.1000000   0.0500000      2.00      -1
# exchange        2
   0     -1.00 315.8560355      312.70   0.0200000   0.0200000      0.00      -1
   1     -1.00 284.9371973      373.51   0.0000000   0.0000000      1.00      -1
   2     -1.00 383.2563608      296.93   0.1000000   0.1000000      0.00      -1
   3     -1.00 307.1145845      181.45   0.0500000   0.0500000      1.00      -1
……
```


6.2. MSESの入力パラメタ

MM input

ncg = 10 (not CGMD, default)

kmmcg: coefficient for MM/CG constraint

$$V_{MM/CG} = \frac{1}{2} k_{MMCG} \sum_{i,j} \left[d_{\mathrm{MM,ij}} - d_{\mathrm{CG,ij}} \right]^2$$

nemmcg = 0 (all the distances for nonbonded residue pairs, default)

1 (using lstmcg by two-domain description)

```
ex.  2            : number of constraints is 2
     31  60  61  126   :between 31-60 and 61-126
     61  126 127 164 :between 61-126 and 127-164
```

2 (using lstmcg by two-residue-number description)

```
ex. 2000          : for 2000 residue pairs
     31  61       : between 31 and 61
     ……
```

lstmcg: filename for constraint list, nemmcg = 1, 2)

numexchg: number of replica exchanges

(total timestep is nstlim×numexchg)

CG input

ncg = 11 (elastic network model: -ref1)

13 (plastic network model: -ref1,-ref2)

14 (mixed elastic network model: -ref1,-ref2)

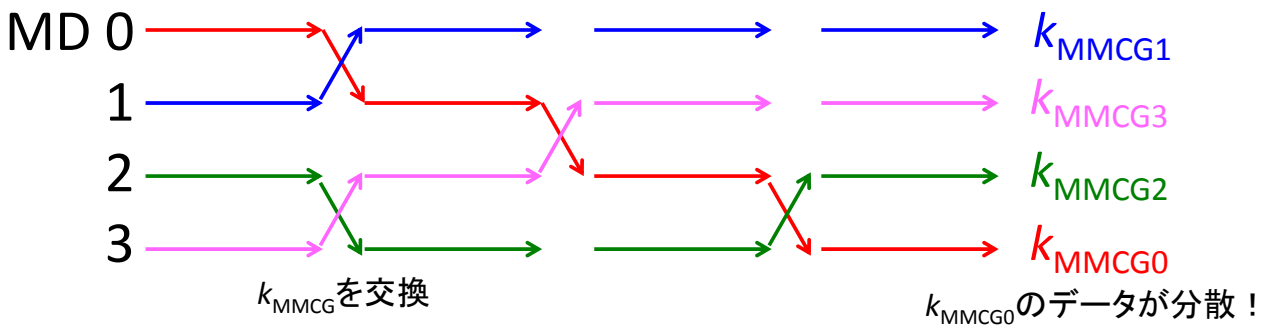
15 (microscopic plastic network model : -ref1,-ref2)

6.3. MSES結果の解析

MSESで得られた全原子構造アンサンブルから自由エネルギー地形を計算し、single-copy MDの結果と比較する

1. ptrajで各レプリカのトラジェクトリからunbiasedな（k_{MMCG} = 0）トラジェクトリを抽出、水素結合距離を計算する
2. 結果をプロット、自由エネルギー地形を計算

k_{MMCG} = 0での水素結合距離データ

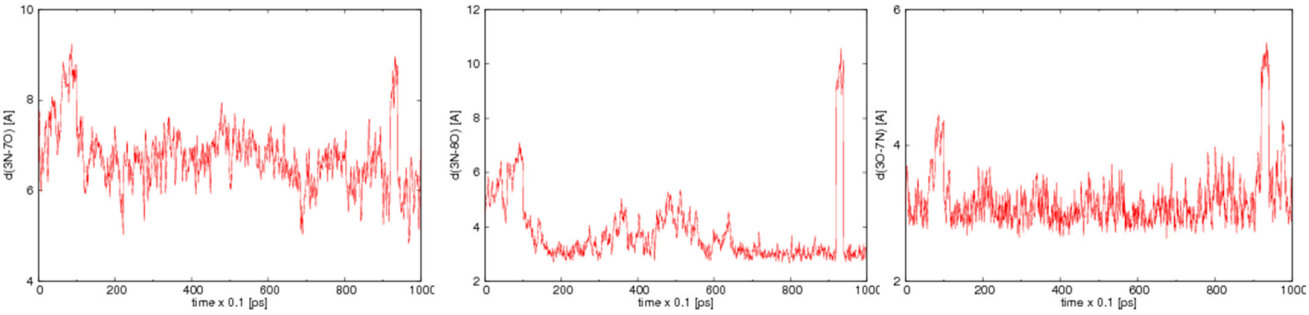


ptraj.sh

```
#!/bin/tcsh

setenv AMBERHOME /home/islim/mu2lib-k/amber12
$AMBERHOME/bin/ptraj 1uao.top << EOF
reference 1uao.crd
trajin mdcrd.000 remdtraj remdtrajtemp 0.0000000
rms reference out rmsd.dat :2-9@CA
distance x :3@N :7@O out dist3N7O.dat
distance y :3@N :8@O out dist3N8O.dat
distance y :3@O :7@N out dist3O7N.dat
go
EOF
```

実行、gnuplotで見る (またはpngファイル作成)

```
$ ./ptraj.sh

$ gnuplot
Terminal type set to 'x11'
gnuplot> plot 'dist3N7O.dat' w l
gnuplot> plot 'dist3N8O.dat' w l
gnuplot> plot 'dist3O7N.dat' w l
gnuplot> quit

$ gnuplot plot1.gnu;convert dist3N7O.eps dist3N7O.png
$ gnuplot plot2.gnu;convert dist3N8O.eps dist3N8O.png
$ gnuplot plot3.gnu;convert dist3O7N.eps dist3O7N.png
```



自由エネルギー地形の計算

fes.sh

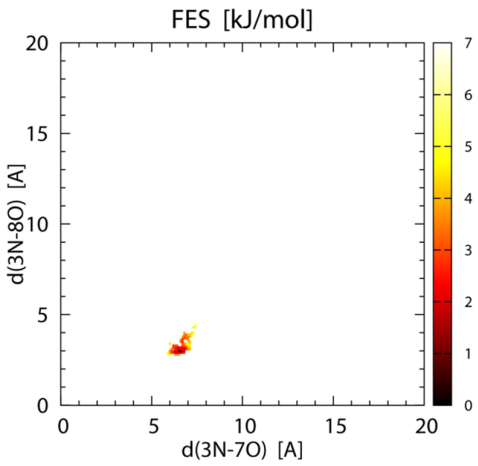
```
#!/bin/tcsh

g77 -o calfes calfes.f
###  xmin xmax ymin ymax nbinx nbiny
echo " 0.0 20.0 0.0 20.0 200 200" > fort.10
##  temperature
echo " 300.0" >> fort.10

¥rm -f fort.11 fort.12
ln -s dist3N7O.dat fort.11
ln -s dist3N8O.dat fort.12
./calfes > fes.dat
```

実行 (map.pngが作成される)

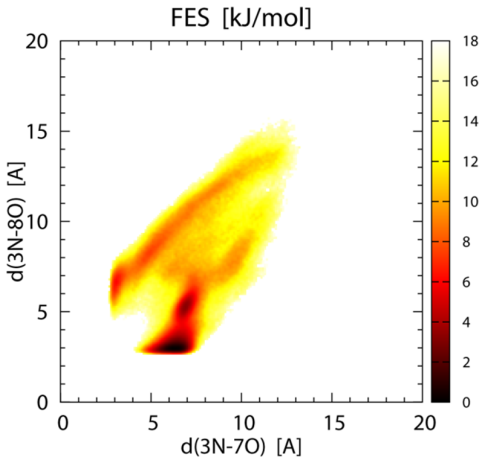
```
$ ./fes.sh
$ gnuplot map.gnu; convert map.eps map.png
```

100-nsシミュレーションした結果を参照 (map100ns.png)

```
$ ./fes_100ns.sh
$ gnuplot map_100ns.gnu; convert map100ns.eps map100ns.png
```



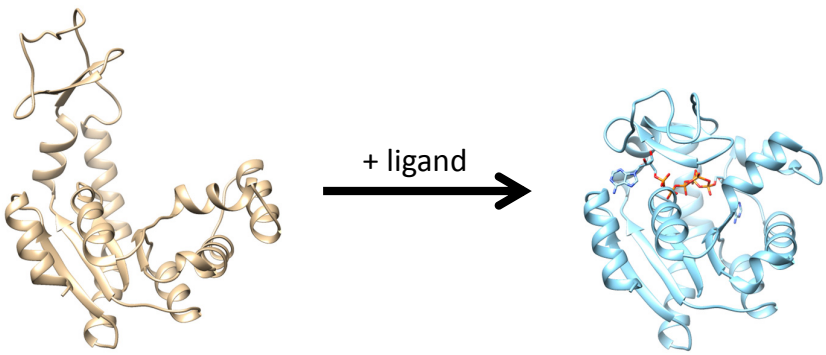
今回のシミュレーション（100 ps）でのFES



100-ns シミュレーションでのFES

7. ストリング法の概要（1）

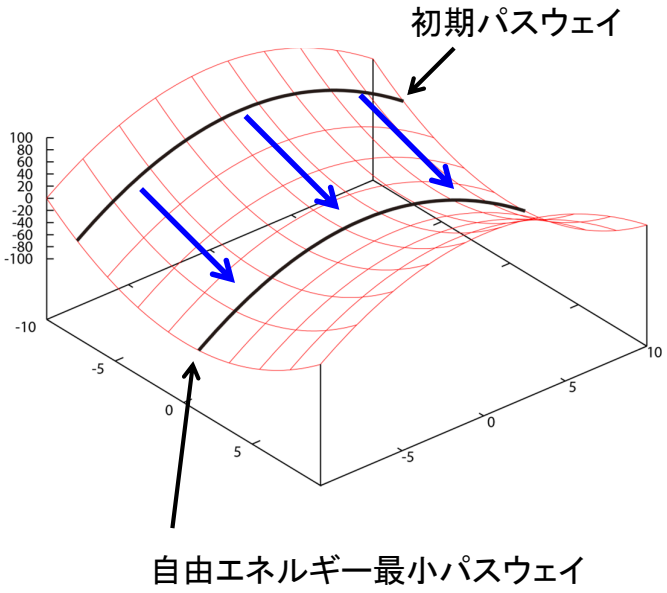
- 最も可能性の高い、立体構造変化過程（パスウェイ）を求める



リガンド非結合構造　　リガンド結合構造

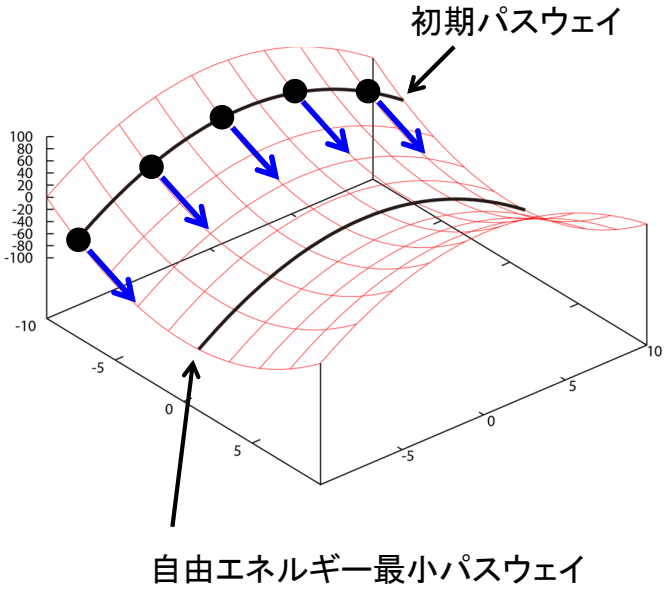
7. ストリング法の概要（2）

- 最も可能性の高い立体構造変化パスウェイ
  ||
  自由エネルギー最小パスウェイ
- 初期パスウェイを自由エネルギーの勾配に従って最適化すればよい



7. ストリング法の概要（3）

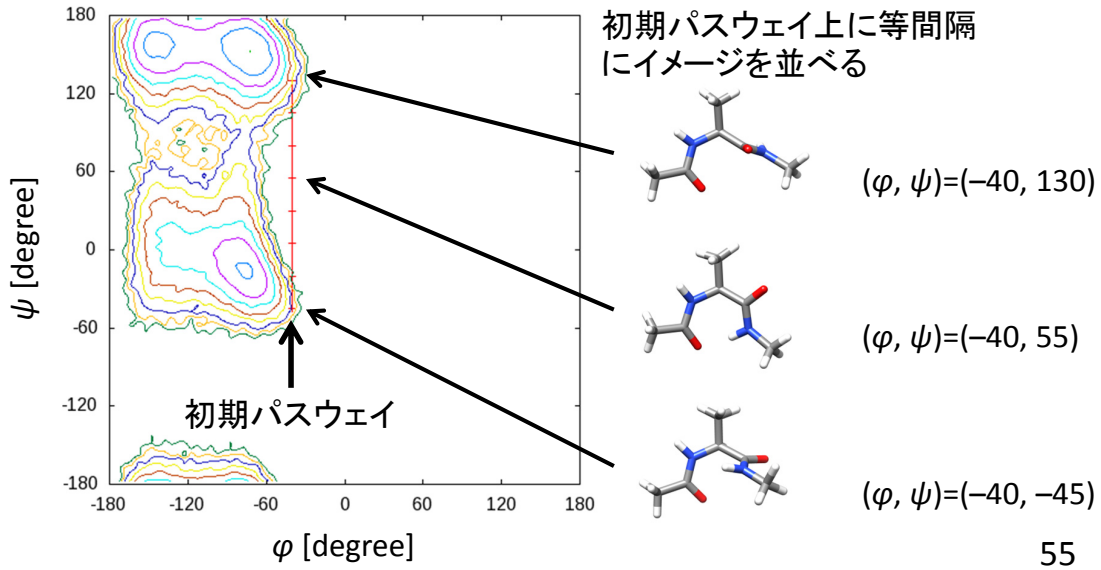
- パスウェイを離散化
  ||
  パスウェイ上に構造が少しずつ異なるイメージを配置
- 各イメージで自由エネルギー勾配のパスウェイに垂直な成分を計算し、その向きにイメージを移動する



Maragliano, Vanden-Eijnden, *Chem.Phys.Lett.* **446,** 182 (2007).

8. ストリング法実習（1）

- Alanine dipeptideの構造変化における自由エネルギー最小パスウェイを求める



8. ストリング法実習（2）

1. ストリング法の実行
   - 16コピー（8イメージ×2）、100 ps定温定圧MD
2. 自由エネルギープロファイル（PMF）の計算
   - 16コピー（8イメージ×2）、50 ps定温定圧MD
3. 結果の解析
   - パスウェイのプロット
   - 自由エネルギープロファイルのプロット

8. ストリング法実習（3）

- 作業ディレクトリに移動

```
> cd ~/tutorial/string
```

- ディレクトリ構成
  - run: ストリング法シミュレーション実行
  - pmf: 自由エネルギープロファイル（Potential of mean force）計算用シミュレーション実行
  - analysis: 解析

8.1. ストリング法の実行（1）

- runディレクトリに移動

```
> cd run
```

- 入力ファイルを作成

```
> ./generate.pl
```

- 結果の確認

```
> ls
generate.pl  md02x.in  md04y.in  md07x.in   rst03.dat  run.sh
md00x.in     md02y.in  md05x.in  md07y.in   rst04.dat
md00y.in     md03x.in  md05y.in  rst00.dat  rst05.dat
md01x.in     md03y.in  md06x.in  rst01.dat  rst06.dat
md01y.in     md04x.in  md06y.in  rst02.dat  rst07.dat
```

8.1. ストリング法の実行（2）

- md00x.in

```
molecular dynamics run
 &cntrl
    imin=0, nmropt=1,
    ntx=5, irest=1, ntrx=1,
    ntxo=1, ntpr=50, ntwr=50, ntwx=50,
    ioutfm=1,
    ntr=0,
    maxcyc=2000, ncyc=1000, ntmin=1,
    nstlim=50000, nscm=500, t=0.0,
    dt=0.002,
    ntt=3, temp0=300.0, tempi=300.0,
    gamma_ln=2.0, ig=7374,
    ntp=1, pres0=1.0, taup=1.0,
    ntc=2, tol=1.0e-6,
    ivcap=0, fcap=1.5,
    ntf=2, ntb=2, cut=8.0, igb=0,
 /
DISANG=rst00.dat
```

- rst00.dat

```
 &rst
    iat=5, 7, 9, 15,
    r1=-1000.0, r2=-40, r3=-40,
r4=1000.0,
    rk2=1000.0, rk3=1000.0,
 /
 &rst
    iat=7, 9, 15, 17,
    r1=-1000.0, r2=130, r3=130,
r4=1000.0,
    rk2=1000.0, rk3=1000.0,
 /
```


8.1. ストリング法の実行（3）

- run.sh

```
#!/bin/sh
#------pjsub options ------#
#PJM -L "rscgrp=small"
#PJM -L "node=4"
#PJM --mpi "proc=16"
#PJM -L "elapse=30:00"
#PJM -j
#------Program Execution --------#
export OMP_NUM_THREADS 4
mpiexec $HOME/tstring/tstring -O ¥
  -ncopy 16 ¥
  -gamma 50.0 ¥
  -rearrange_freq 50 ¥
  -i md%02d%c.in ¥
  -o md%02d%c.out ¥
  -p ../data/leap.top ¥
  -c ../data/md%02d.restrt ¥
  -r md%02d%c.restrt ¥
  -rst_out md%02d%c.rst ¥
  -x md%02d%c.crd ¥
  -extra md%02d%c.extra
```

- 4ノード使用
- –ncopy: コピー数（8イメージ×2）
- –gamma: 摩擦係数（大きいほど経路の移動が遅い）
- –rearrange_freq: 経路上の点の再配置の頻度
- 以下によりジョブを投入

```
> pjsub ./run.sh
```

8.1. ストリング法の実行（4）

- 計算が終了したらanalysisディレクトリに移動

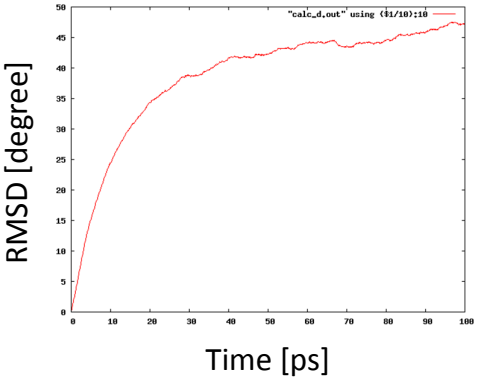
```
> Cd ../analysis
```

- イメージの移動度（RMSD）をプロットする

```
> ./calc_d.pl
> ./calc_d.gp
```

- calc_d.pngを表示する
  - 収束の度合いを確認

$$\mathrm{RMSD}(t) = \sqrt{\frac{1}{R}\sum_{p=1}^{R}\left|\mathbf{z}_p(t) - \mathbf{z}_p(0)\right|^2}$$



8.2. PMFの計算（1）

- pmfディレクトリに移動

```
> cd ../pmf
```

- 入力ファイルを作成（ストリング法の計算が終了してから実行すること）

```
> ./generate.pl
```

- 結果の確認

```
> ls
generate.pl  md02x.in  md04y.in  md07x.in   rst03.dat  run.sh
md00x.in     md02y.in  md05x.in  md07y.in   rst04.dat
md00y.in     md03x.in  md05y.in  rst00.dat  rst05.dat
md01x.in     md03y.in  md06x.in  rst01.dat  rst06.dat
md01y.in     md04x.in  md06y.in  rst02.dat  rst07.dat
```

8.2. PMFの計算（2）

- md00x.in

```
molecular dynamics run
 &cntrl
    imin=0, nmropt=1,
    ntx=5, irest=1, ntrx=1,
    ntxo=1, ntpr=50, ntwr=50, ntwx=5,
    ioutfm=1,
    ntr=0,
    maxcyc=2000, ncyc=1000, ntmin=1,
    nstlim=25000, nscm=500, t=0.0,
    dt=0.002,
    ntt=3, temp0=300.0, tempi=300.0,
    gamma_ln=2.0, ig=7374,
    ntp=1, pres0=1.0, taup=1.0,
    ntc=2, tol=1.0e-6,
    ivcap=0, fcap=1.5,
    ntf=2, ntb=2, cut=8.0, igb=0,
 /
DISANG=rst00.dat
```

- rst00.dat

```
&rst
    iat=5, 7, 9, 15,
    r1=-1000.0, r2=-69.7258219870418,
    r3=-69.7258219870418, r4=1000.0,
    rk2=1000.0, rk3=1000.0,
 /
 &rst
    iat=7, 9, 15, 17,
    r1=-1000.0, r2=152.750175587423,
    r3=152.750175587423, r4=1000.0,
    rk2=1000.0, rk3=1000.0,
 /
```


8.2. PMFの計算（3）

- run.sh

```
#!/bin/sh
#------pjsub options ------#
#PJM -L "rscgrp=small"
#PJM -L "node=4"
#PJM --mpi "proc=16"
#PJM -L "elapse=30:00"
#PJM -j
#------Program Execution --------#
export OMP_NUM_THREADS 4
mpiexec $HOME/tstring/tstring -O ¥
  -ncopy 16 ¥
  -gamma 0.0 ¥
  -rearrange_freq 50 ¥
  -i md%02d%c.in ¥
  -o md%02d%c.out ¥
  -p ../data/leap.top ¥
  -c ../run/md%02d%c.restrt ¥
  -r md%02d%c.restrt ¥
  -rst_out md%02d%c.rst ¥
  -x md%02d%c.crd ¥
  -extra md%02d%c.extra
```

- 4ノード使用
- –ncopy: コピー数（8イメージ×2）
- –gamma: 0.0に設定する（イメージが固定される）
- –rearrange_freq: イメージの座標書き出しの頻度
- 以下によりジョブを投入

```
> pjsub ./run.sh
```

8.3. 結果の解析

1. パスウェイのプロット
2. 自由エネルギープロファイルのプロット
3. 立体構造の確認

8.3.1.パスウェイのプロット

- analysisディレクトリに移動し、以下を実行

```
> cd ../analysis
> ./draw_string.pl
```

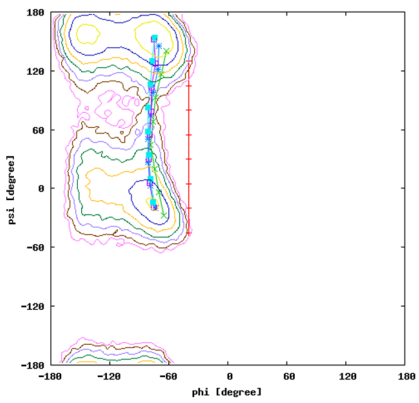
- 別途計算した二面角(φ, ψ)の分布から自由エネルギー曲面の等高線を計算

```
> ./contour.gp
```

- 結果のプロット

```
> ./draw_string.gp
```

→string.pngが生成される



8.3.2. 自由エネルギープロファイル

- 以下を実行

```
> ./pmf.pl
```

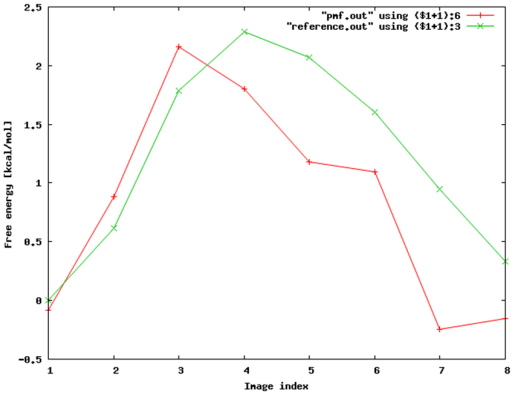
- 別途計算した二面角(φ, ψ)のトラジェクトリから、自由エネルギープロファイルを計算

```
> ./reference.pl
```

- 結果を比較する

```
> ./pmf.gp
```

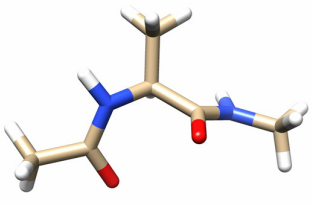
→pmf.pngが生成される



8.3.3. 立体構造の確認

- ambpdb.sh

```
#!/bin/tcsh
setenv AMBERHOME /home/islim/mu2lib-k/amber12
if ( ! -e pdb ) then
  mkdir pdb
endif
foreach in (../run/*.restrt)
  set out=`basename $in`
  $AMBERHOME/bin/ambpdb -p ../data/leap.top < $in |¥
  head -23 > pdb/$out:r.pdb
end
```



- これを実行すると、最終構造のPDBファイルがpdbディレクトリに生成される
  - UCSF ChimeraやVMD等で立体構造を確認