

CBI学会2017年大会

# GENESISチュートリアル2 自由エネルギー計算 (実習)

理化学研究所 計算科学研究機構  
粒子系生物物理研究チーム

講演者：神谷基司

[motoshi.kamiya@riken.jp](mailto:motoshi.kamiya@riken.jp)

2017年10月5日 タワーホール船堀 (東京)

# 今日の内容

## ■ 15:45 – 17:00 GENESISでの自由エネルギー計算

- 自由エネルギー計算とは?
- アンブレラサンプリング法について
- レプリカ交換アンブレラサンプリング法(実習)
- WHAM法による自由エネルギー計算(実習)

# アンブレラサンプリング法

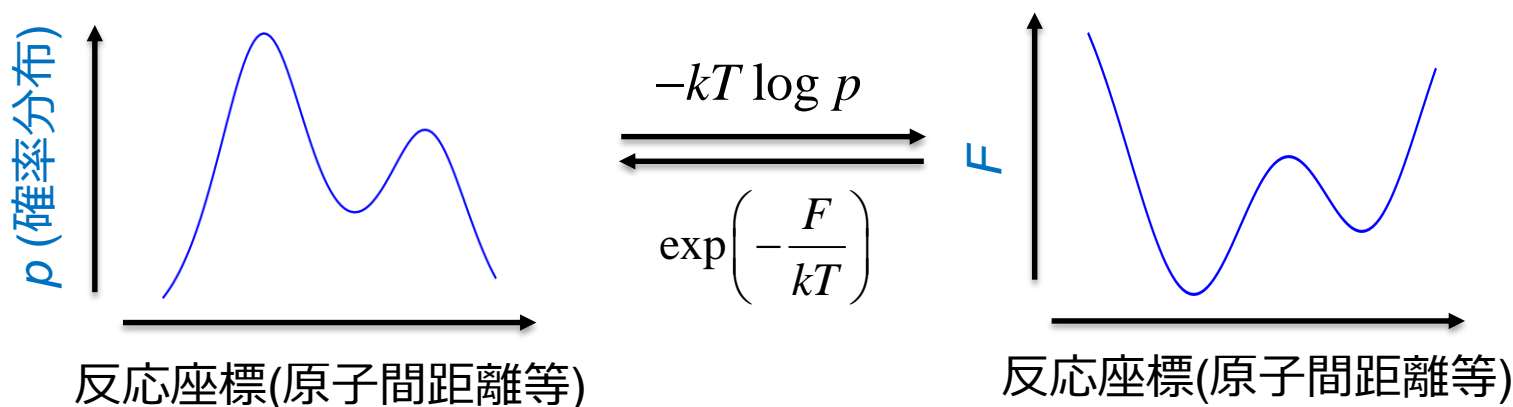
# 自由エネルギー計算

$$F = E - TS$$

.....とかいう式は特に意識する必要ありません  
(してもいいです)

✎ 自由エネルギーは(意外かもしれませんが)直観的に理解できる量でもある

- 結合した状態と結合していない状態の比率
- 何らかの反応座標軸の分布



✎ 自由エネルギーはむしろ現象論的な観測量(っぽいもの)と直接的に関係する  
(エントロピーはもちろんエネルギーよりも直観的に理解しやすい)

# 色々な自由エネルギー計算方法

(より正確には相対的自由エネルギーの計算方法)

## ■ 自由エネルギー摂動法(free energy perturbation)

- 仮想的な状態を間に挿入し、状態間の自由エネルギー差を計算する
- 粒子を生成消滅するような計算も可能
- 結合自由エネルギー計算に良く用いられる(REST/FEP等含む)

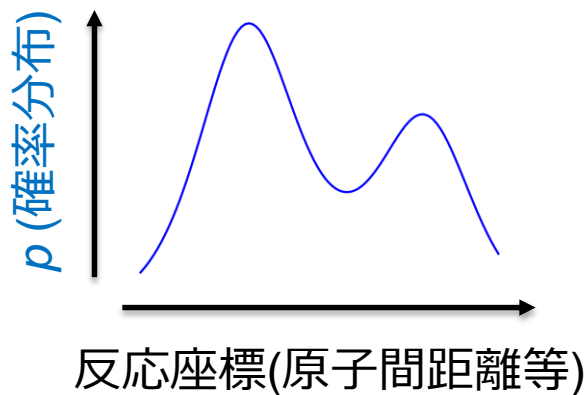
## ■ アンブレラサンプリング(umbrella sampling)

- 拘束ポテンシャルを導入し、反応座標に沿った自由エネルギー変化を計算する  
(反応座標は粒子間距離や角度等を用いる)
- 粒子を生成消滅させるような計算は通常はできない
- 結合ポーズの高精度な推定、改善
- リガンド結合の自由エネルギー障壁の高さ(~反応速度)推定

他にも様々な自由エネルギー計算法が存在する。

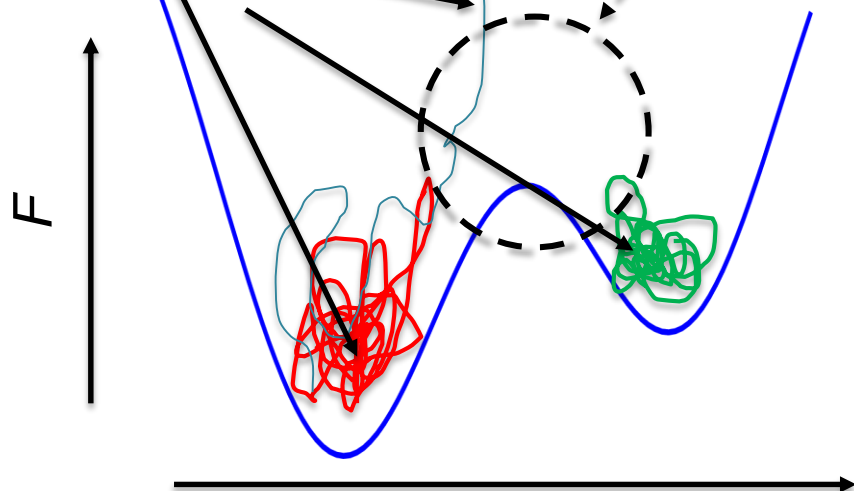
さらには、REST/FEP や REST/REUS 法のように複数組み合わせる場合もある。

# アンブレラサンプリング法(1)



通常のMD計算でこのような分布が得られれば問題ない  
.....が、普通はそううまくいかない

MDトラジェクトリ



不安定領域の構造サンプルはとりにくい

安定性に $\Delta F$ の差がある場合の存在比 =  $\exp\left(-\frac{\Delta F}{kT}\right)$

$\Delta F = 0$  kcal/mol の場合 = 1 : 1

3 kcal/mol の場合 =  $1 : \exp(-3 / 0.6)$   
= 1 : 0.0067

5 kcal/mol の場合 = 1 : 0.00024

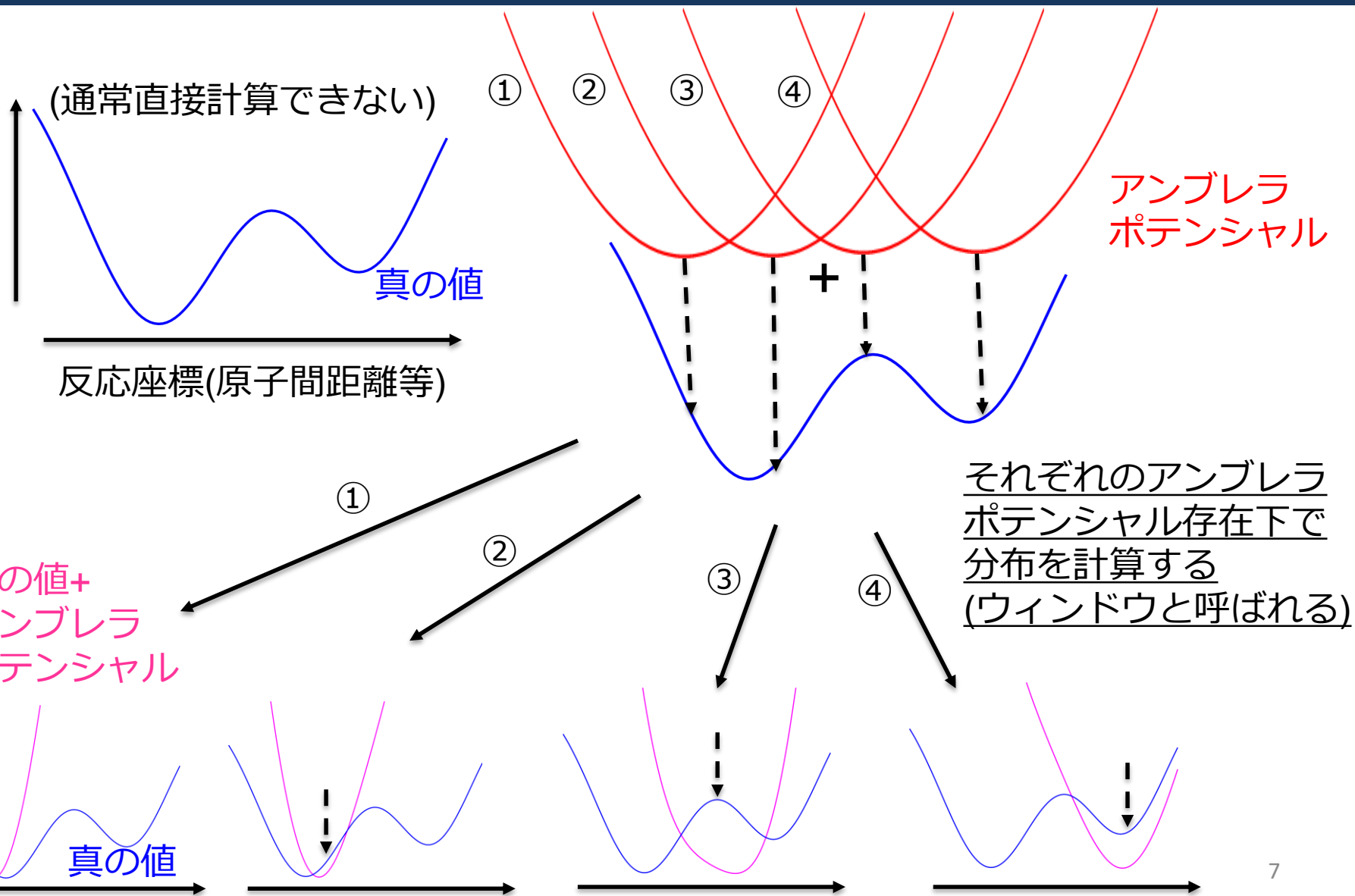
10 kcal/mol の場合 = 1 : 0.000000058

(※ 300 K)

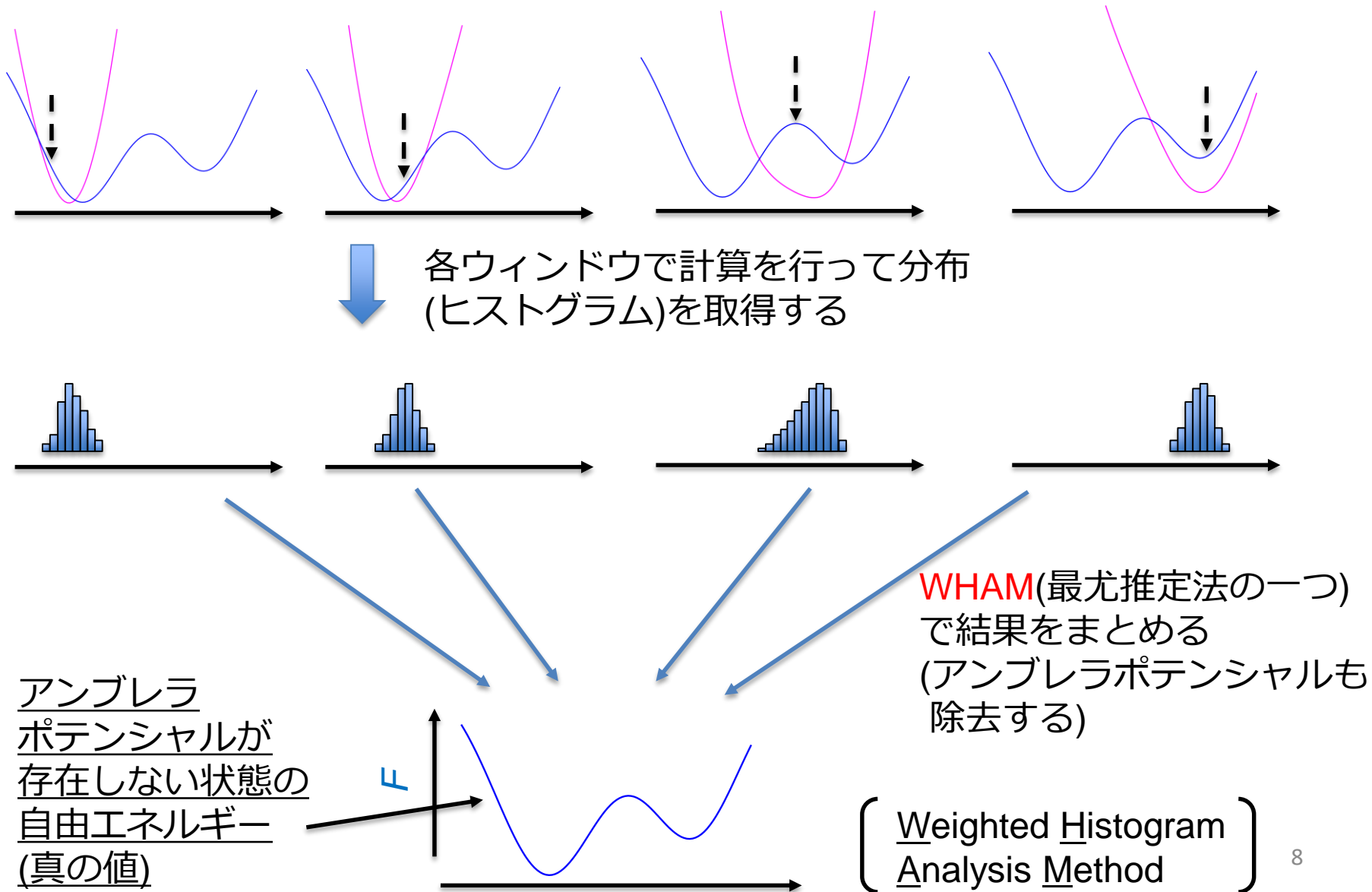


どうにかしたい.....

# アンブレラサンプリング法(2)

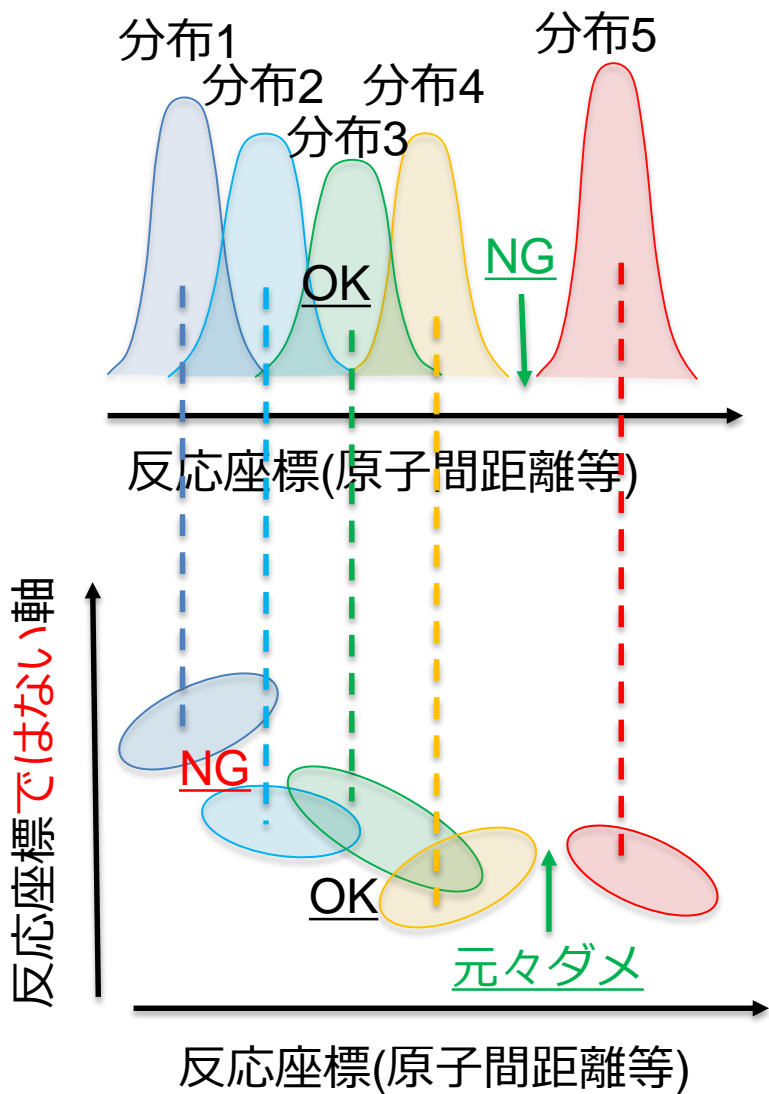


# アンブレラサンプリング法(3)





# アンブレラサンプリング法(4)

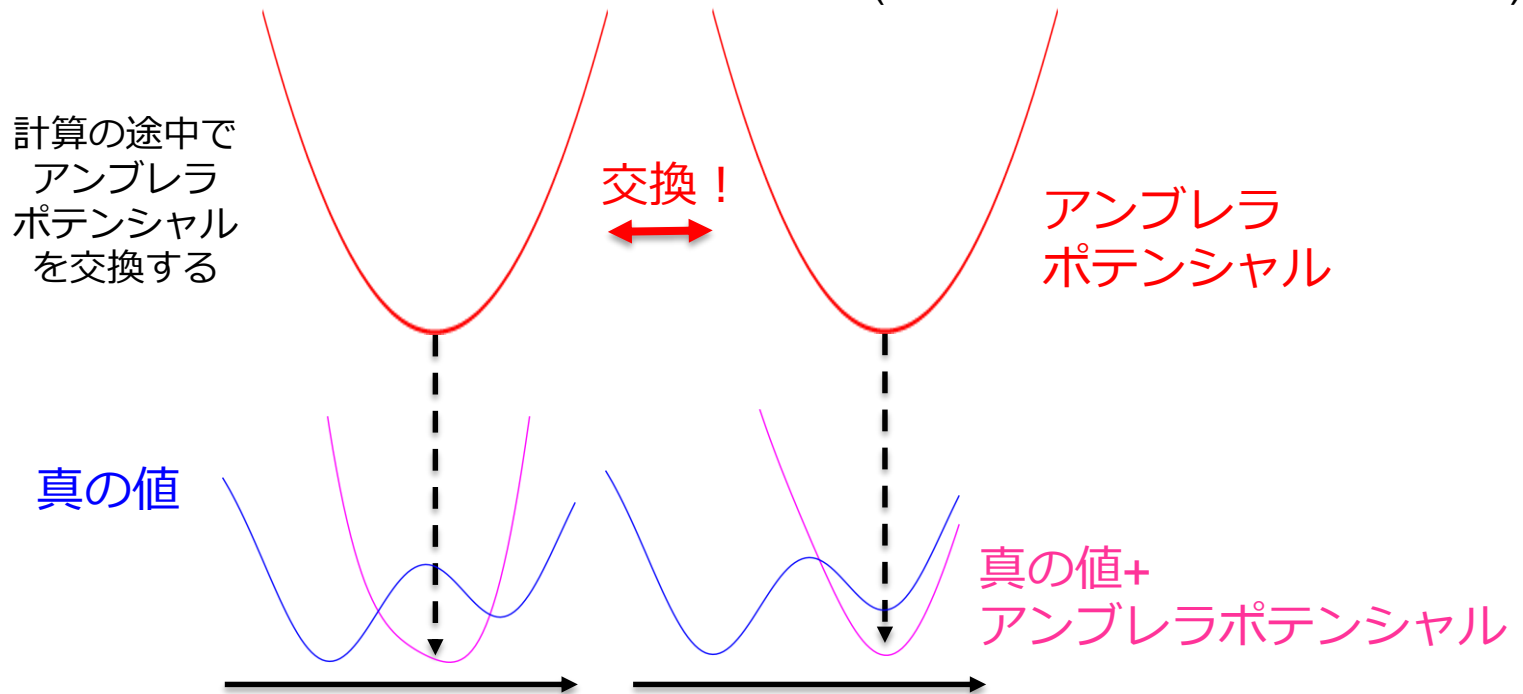


- 分布にはある程度重なりがなければならない
- アンブレラポテンシャルの場所や形の調整
  - 計算を長くする
- 反応座標以外の方向「も」重なりが必須
- 反応座標が状態量である必要性  
= 反応座標がわかれば分布が一意に決まる  
(ものすごく厳密な表現では)
- ↑
- この条件が満たされなくてもWHAMを使えば自由エネルギーは計算できてしまう  
(正しい理由は無い)
- 条件をきちんと満たした正しい計算を行うことは専門家であっても極めて難しい
- (用いる手法によって色々と例外有り)  
(他にも気にすべきことはたくさん有り)

# レプリカ交換アンブレラサンプリング法

# レプリカ交換アンブレラサンプリング法(1)

✎ アンブレラ法の精度を向上させる方法の一つ(計算コストは多少犠牲になる)

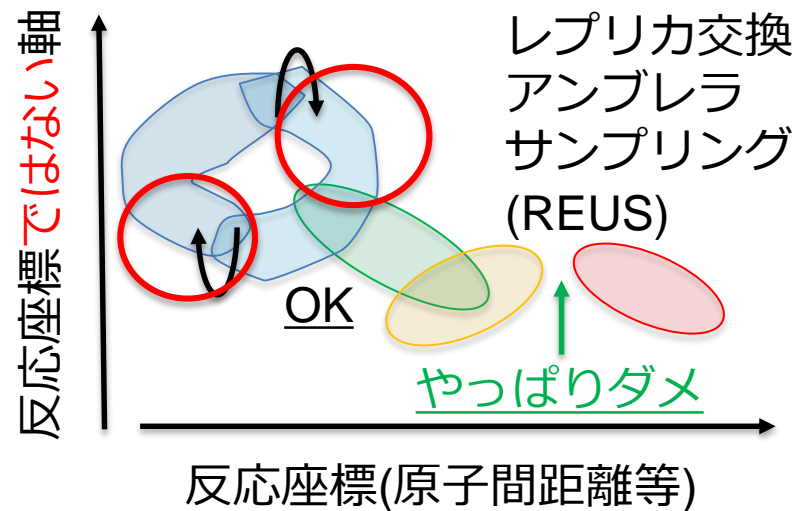
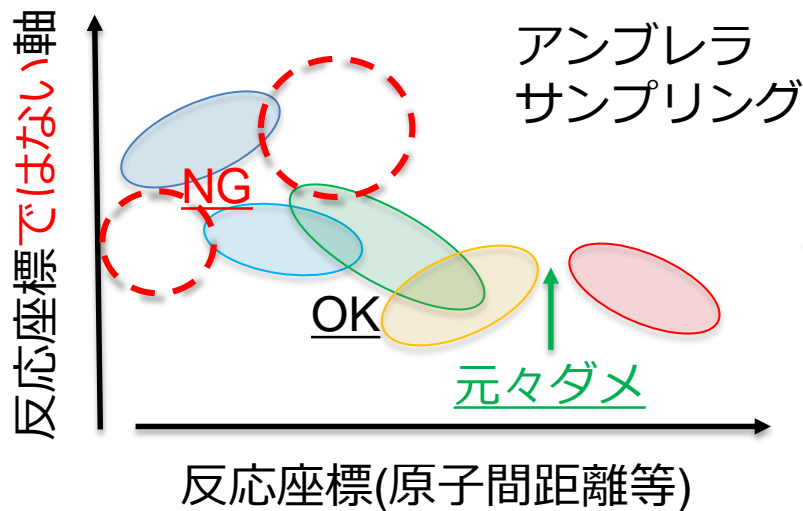


✎ アンブレラポテンシャルの数だけ同時に計算を行う  
交換を伴うために全ての計算が実質的に同期されていなければならない

(通常のアムブレラサンプリングの場合には計算を個別に行える)

(トータルの計算量自体は大きく増えているわけではない)

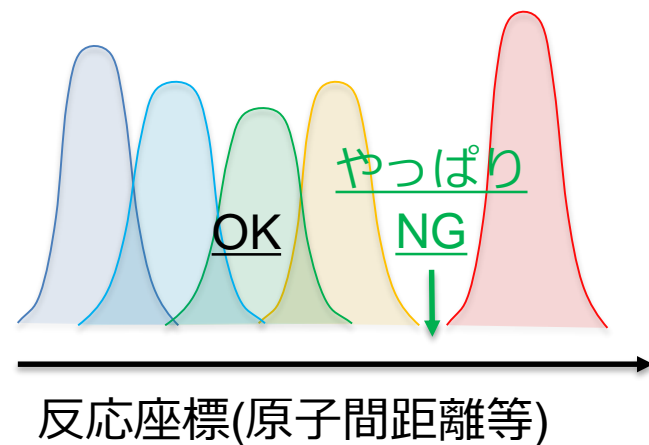
# レプリカ交換アンブレラサンプリング法(2)



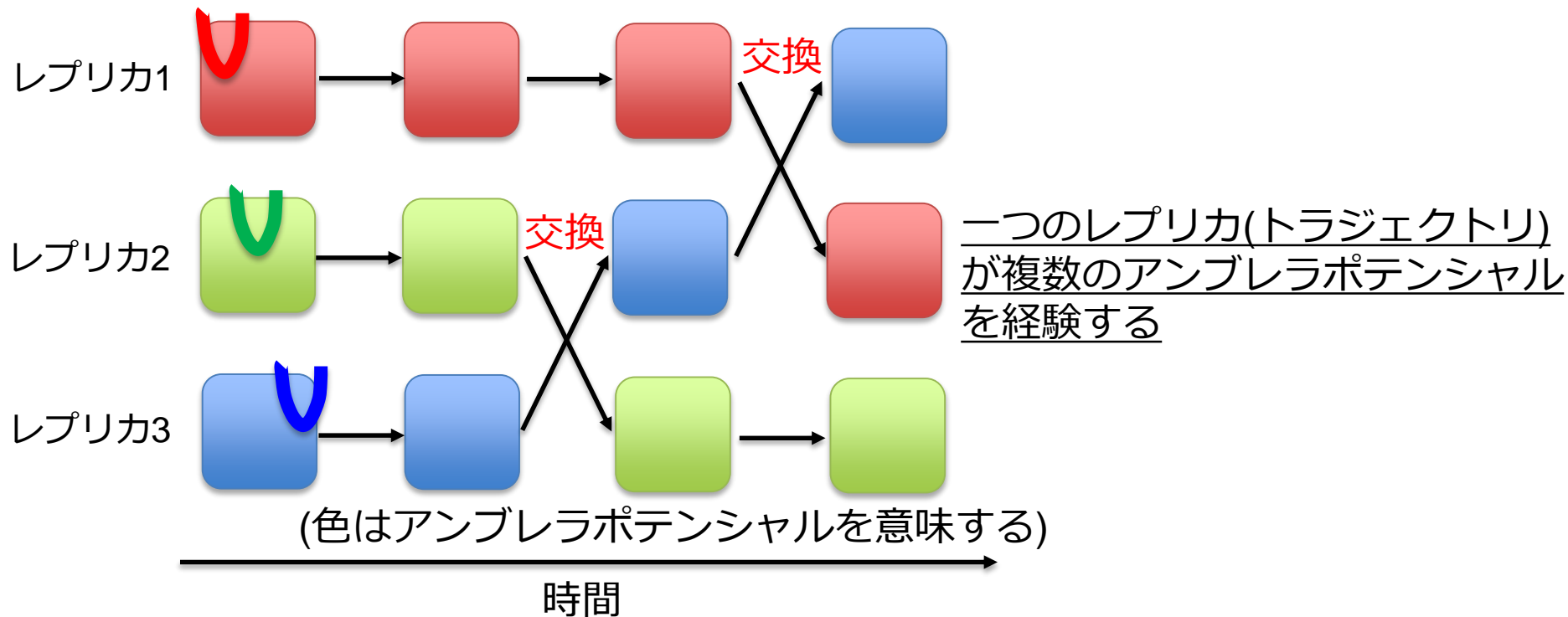
- 交換によってサンプリング空間が拡大する
  - ➡ 前述のめんどくさい問題の改善が期待できる
  - ➡ 様々な安定構造が得られるようになるかも

交換は詳細つり合いを満たす方式で行う  
無条件に交換できるわけではない

反応座標方向の分布に重なりがあり、  
交換が行われていれば良いサンプリングが  
できると期待できる



# レプリカ交換アンブレラサンプリング法(3)



通常のアンブレラ法を使うよりも広い範囲の構造探索が可能

リガンド結合の場合では、一旦脱離、再び結合するような構造変化が探索できる(通常のアンブレラ法や自由エネルギー摂動法では難しい)

# 実習(レプリカ交換アンブレラサンプリング)

# 計算の流れ

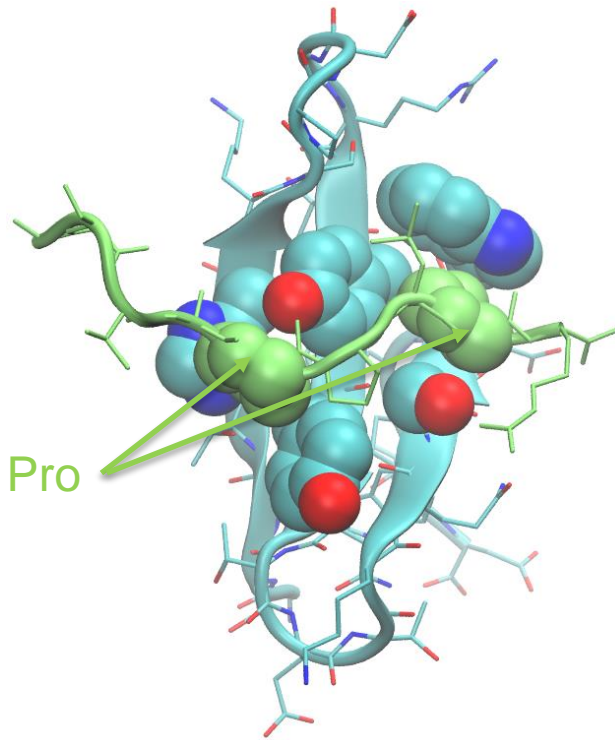
- サンプル系の紹介
- 反応座標の設定
- (システムのセットアップ(省略))
- レプリカ交換アンブレラサンプリング(平衡化)
- レプリカ交換アンブレラサンプリング(サンプリング)
- トラジェクトリの変換
- トラジェクトリから距離の計算
- 自由エネルギー計算
- (時間があれば)より長いトラジェクトリを使った解析  
(赤字部分が実習)

GENESIS ウェブサイト(ただし英語)にもレプリカ交換アンブレラサンプリングのチュートリアルがあります

<http://www.aics.riken.jp/labs/cbrt/tutorial/remd-tutorials/tutorial-2-2/>

今回扱う系よりもシンプルな系でのサンプルです  
(計算ステップ数が多いので適宜調節してください)

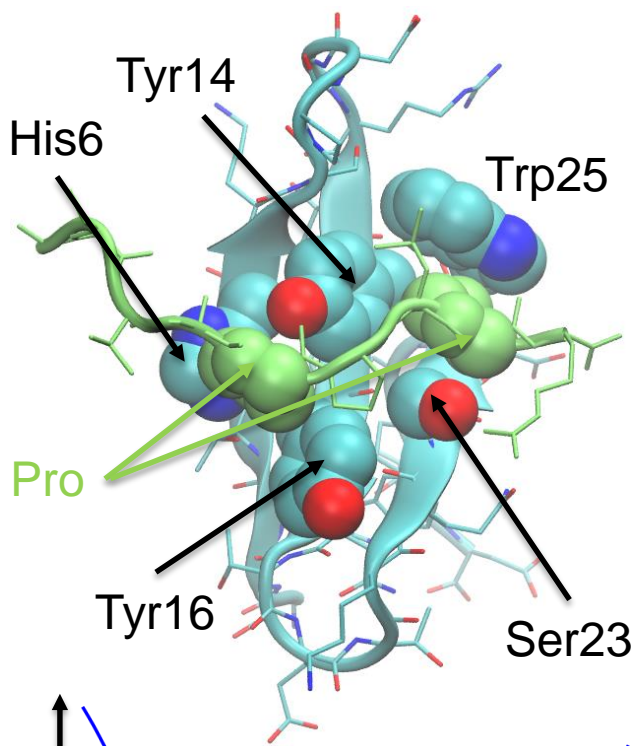
# 簡単なサンプル系



- ✎ PDB ID: 2DYF
- ✎ FBP11/HYPAに含まれるWWドメインの一つ  
Proリッチなペプチドと結合する
- ✎ GSTAPPLPRという配列を持つペプチド  
が結合している
- ✎ 全体で40残基程度で非常に小さい  
(今回の設定では溶媒の水をいれても  
10,000原子程度)



# 結合の反応座標の設定(1)



2 個のProが溝にはまり込むような形になっている

右: Tyr14, Ser23, Trp25 の間 => 結合サイト 1

左: His6, Tyr14, Tyr16 の間 => 結合サイト 2

両方の結合サイトを独立に扱うこともできるが、今回は二つをまとめて扱う



Pro(右)-結合サイト 1 の距離 +  
Pro(左)-結合サイト 2 の距離

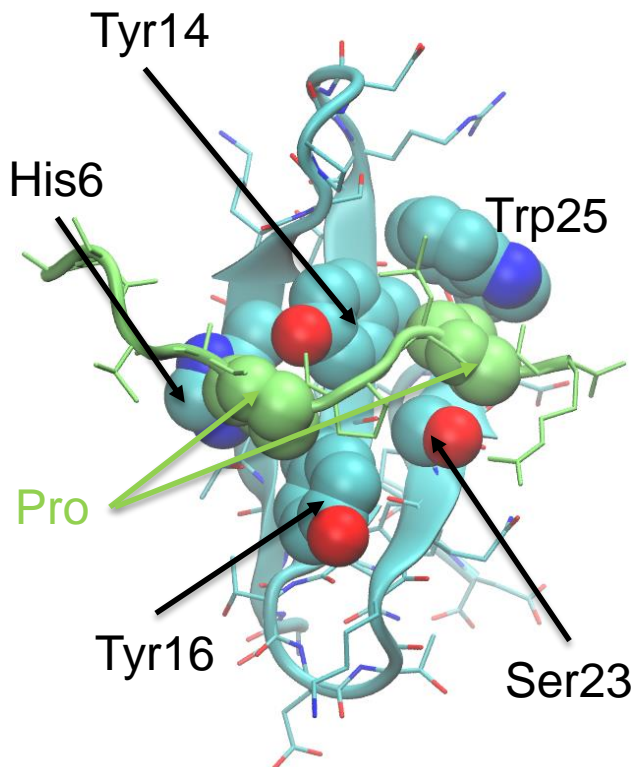
を反応座標として用いる

反応座標(原子間距離等)

コレ

独立に扱う場合に比べて計算量的に大きく有利  
ただし、注意するべき点が多少増える

# 結合の反応座標設定(2)



Tyr14 C $\gamma$ , Trp25 C $\epsilon$ 2, Ser23 C $\beta$  の重心  
とPro(右)の 5 員環炭素の重心の距離

->group2

->group1

+

His6 C $\gamma$ , Tyr14 C $\zeta$ , Tyr16 C $\gamma$  の重心  
とPro(左)の 5 員環炭素の重心の距離

->group4

->group3

```
[RESTRAINTS]
nfunctions          = 1
function1           = DISTMASS
select_index1       = 1 2 3 4
constant1           = ...
reference1           = ...
```

アンブレラポテンシャル:

$$k(r - r_0)^2$$

constant

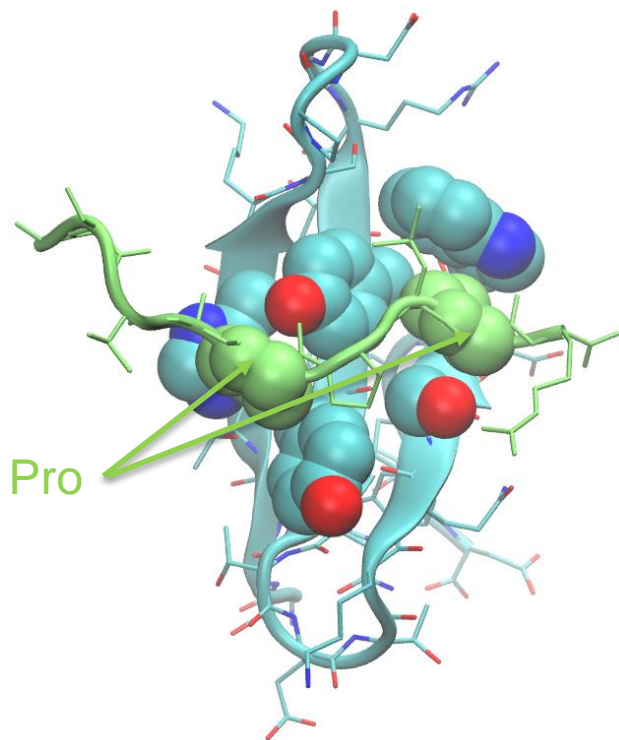
reference

[SELECTION]

```
group1 = rno:38 and (an:CG or an:CB or an:CD or an:CA)
group2 = (rno:25 and an:CE2) or (rno:14 and an:CG) or ¥
        (rno:23 and an:CB)
group3 = rno:35 and (an:CG or an:CB or an:CD or an:CA)
group4 = (rno:6 and an:CG) or (rno:14 and an:CZ) or ¥
        (rno:16 and an:CG)
```

(今回関わる原子  
は全て炭素なので  
重心でも中心でも  
同じ。DISTMASS  
のかわりにDISTも可)

# システムセットアップ(実習は省略)



(AMBER形式; 実習では時間の関係で省略)

- ✎ NMR構造の一つを取り出す
- ✎ タンパク質周囲に水を加えた構造データ(トポロジー)の作成
- ✎ 構造最適化  
(`contact_check` は gpu 未対応なので注意  
外すか cpu だけを使うようにする)
- ✎ MDによる平衡化  
(粒子数体積温度一定条件(NVT),  
粒子数圧力温度一定条件(NPT))

(ここまでの平衡化は通常のMDと同様、インプットファイル等は置いてあるので参考にしてください)

# 実習(レプリカ交換アンブレラサンプリング ; 平衡化)



計算の実行(spdyn)

```
% cd ~/Tutorial_2/3_reus
% sh run_reus_eq.sh &
```

時間等の制限により  
今回は 4 レプリカだけ



ログの確認

```
メインのログファイル(インプットの処理)
% less reus0.log
各レプリカのログファイル(一番目のレプリカ)
% less reus0_1.log
```

初期構造は 4 つのレプリカ  
全てが同じ構造を使う

reus0.log (抜粋):

```
[STEP4] Compute Single Point Energy for Molecules

          STEP          BOND          ANGLE ...
-----
              0          122.4199          314.7267 ...

[STEP5] Perform Replica-Exchange MD Simulation

[STEP6] Deallocate Arrays

Output_Time> Averaged timer profile (Min, Max)
total time      =      98.154
setup           =       5.117
```

reus0\_1.log (抜粋):

INFO:	STEP	TIME	TOTAL_ENE ...
INFO:	100	0.1000	-25471.7400 ...
INFO:	200	0.2000	-25481.7864 ...
INFO:	300	0.3000	-25497.9852 ...
INFO:	400	0.4000	-25498.1773 ...
INFO:	500	0.5000	-25473.3131 ...
INFO:	600	0.6000	-25424.1691 ...
INFO:	700	0.7000	-25450.2135 ...
INFO:	800	0.8000	-25471.3453 ...

# 実習(レプリカ交換アンブレラサンプリング ; 平衡化)

reus0.inp: (spdyn用インプットファイルの抜粋と説明)

```
[OUTPUT]
logfile      = reus0_{}.log    # -> {} には自動的にレプリカIDが入る
rstfile      = reus0_{}.rst    # ファイルパスの ~ (チルダ)はintel fortran
dcdfile      = reus0_{}.dcd    # ならホームディレクトリに置換されるが、
                                # gfortran では置換されない(バージョン依存あるかも)

[REMD]
dimension    = 1
exchange_period = 0            # 交換頻度; 0の場合は交換しない
iseed        = 4398237        # 交換につかう乱数の種

type1        = RESTRAINT      # レプリカ交換アンブレラサンプリング
nreplica1    = 4              # 4 レプリカを用いる
rest_function1 = 1            # 下の 1 番の restraint を用いる

[RESTRAINTS]
nfunctions   = 1
function1    = DISTMASS       # 今回は DIST でも問題無い(はず)
select_index1 = 1 2 3 4       # [SELECTION]のgroupに対応
constant1    = 1.0 1.0 1.0 1.0 # 力の定数 k; レプリカ数だけ並べる
reference1   = 6.0 7.0 8.0 9.0 # ポテンシャルの中心  $r_0$ ; レプリカ数並べる
```

 レプリカあたりのプロセス数は自動で設定される(合計/レプリカ数; 今回 16 / 4)

 プロセス数は必ず均等。各レプリカには最低4プロセス必要

(OpenMPは適宜) (GPUはあってもなくても可)

# 実習(レプリカ交換アンブレラサンプリング)

## 計算の実行(spdyn)

```
% cd ~/Tutorial_2/3_reus
% sh run_reus_prod.sh &
```

## ログの確認

メインのログファイル  
(インプット処理、**レプリカ交換確率表示等**)  
% less reus1.log  
各レプリカのログファイル(一番目のレプリカ)  
% less reus1\_1.log  
**レプリカの状態(一番目のレプリカ)**  
% less reus1\_1.rem

```
[RESTRAINTS]
nfunctions      = 1
function1       = DISTMASS
select_index1   = 1 2 3 4
constant1       = 1.0 1.0 1.0 1.0
reference1      = 6.0 7.0 8.0 9.0
```

1 2 3 4

reus1\_1.rem (抜粋):      reus1\_2.rem (抜粋):

0	1	0	2
1000	1	1000	2
2000	1	2000	2
3000	1	3000	2
4000	1	4000	2
5000	1	5000	2
6000	1	6000	2
7000	1	7000	3
8000	1	8000	3
9000	1	9000	3
10000	1	10000	3
11000	1	11000	3
12000	1	12000	3
13000	1	13000	3
14000	1	14000	3
15000	1	15000	3
16000	2	16000	3
17000	2	17000	3
18000	2	18000	3
19000	3	19000	2

ステップ数

RESTRAINTS の順番に対応

# 実習(レプリカ交換アンブレラサンプリング)

reus1.inp: (抜粋)

```
[OUTPUT]
logfile          = reus1_{}.log      # -> {} には自動的にレプリカIDが入る
remfile          = reus1_{}.rem      # 現在のパラメータを表すトラジェクトリ
rstfile          = reus1_{}.rst
dcdfile          = reus1_{}.dcd

[REMD]
dimension        = 1
exchange_period  = 1000              # 交換頻度; 1000 ステップに一回
iseed            = 4398237           # 交換につかう乱数の種
                                     # 継続計算時は無視されます
type1            = RESTRAINT        # レプリカ交換アンブレラサンプリング
nreplica1        = 4                # 4 レプリカを用いる
rest_function1   = 1                # 下の 1 番の restraint を用いる

[RESTRAINTS]
nfunctions        = 1
function1         = DISTMASS
select_index1     = 1 2 3 4
constant1         = 1.0 1.0 1.0 1.0 # 力の定数 k; レプリカ数だけ並べる
reference1        = 6.0 7.0 8.0 9.0 # ポテンシャルの中心 r0; レプリカ数並べる
```

# 実習(レプリカ交換アンブレラサンプリング)

reus1.log (抜粋):

```
REMD> Step:      49000  Dimension:      1  ExchangePattern:      2
      Replica      ExchangeTrial      AcceptanceRatio      Before      After
      1          1 >      0  N          0 /      0          1          1
      2          3 >      2  A          11 /      25         3          2
      3          2 >      3  A          11 /      25         2          3
      4          4 >      0  N          0 /      0          4          4

Parameter      :          1          2          3          4
RepIDtoParmID:          1          2          3          4
ParmIDtoRepID:          1          2          3          4

REMD> Step:      50000  Dimension:      1  ExchangePattern:      1
      Replica      ExchangeTrial      AcceptanceRatio      Before      After
      1          1 >      2  A          10 /      25         1          2
      2          2 >      1  A          10 /      25         2          1
      3          3 >      4  R           0 /      25         3          3
      4          4 >      3  R           0 /      25         4          4

Parameter      :          2          1          3          4
RepIDtoParmID:          2          1          3          4
ParmIDtoRepID:          2          1          3          4
```



平衡化の時とは異なり、レプリカの交換確率が表示される



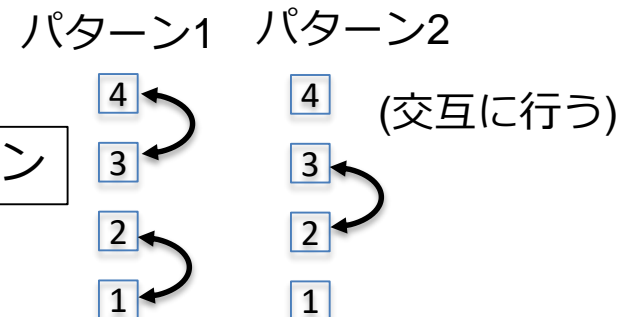
# 実習(レプリカ交換アンブレラサンプリング)

交換試行の結果

A: 交換成功(Accepted)

R: 交換失敗(Rejected)

N: 何もしない



交換のパターン

REMD> Step:	49000	Dimension:	1	ExchangePattern:	2	
Replica	ExchangeTrial			AcceptanceRatio	Before	After
1	1 >	0	N	0 / 0	1	1
2	3 >	2	A	11 / 25	3	2
3	2 >	3	A	11 / 25	2	3
4	4 >	0	N	0 / 0	4	4
Parameter :	1	2	3	4		
RepIDtoParmID:	1	2	3	4		
ParmIDtoRepID:	1	2	3	4		

現在のアンブレラ  
ポテンシャルのID

交換のターゲット(対称)  
0の場合は交換しない

交換確率を表す  
(成功回数 / 試行回数)  
計算をリスタートした際には、  
この回数も引き継がれる

# 実習(トラジェクトリの変換; remd\_convert)

レプリカ交換アンブレラサンプリング計算が終わった段階では、各トラジェクトリは色々なアンブレラポテンシャルでのデータが混じっている

➡ 構造の再配列を行う (同じアンブレラポテンシャルのデータを集める)

➡ 水の部分を取り除いて取り扱いを楽にしておく

## 計算の実行 (remd\_convert)

```
% cd ~/Tutorial_2/4_conv  
% sh conv.sh &
```

## ログの確認

```
ログ確認  
% less conv1.log  
トラジェクトリのファイルサイズ  
% ls -sh
```

conv1.log (抜粋):

```
MD step:      16000  Replica:      1  
Open trajectory file: "../reus/reus1_3.dcd"  
MD step:      17000  Replica:      3  
MD step:      18000  Replica:      3  
MD step:      19000  Replica:      3  
MD step:      20000  Replica:      3  
MD step:      21000  Replica:      3  
MD step:      22000  Replica:      3  
MD step:      23000  Replica:      3  
MD step:      24000  Replica:      3  
MD step:      25000  Replica:      1  
MD step:      26000  Replica:      1  
Open trajectory file: "../reus/reus1_2.dcd"  
MD step:      27000  Replica:      2  
MD step:      28000  Replica:      2
```

# 実習(トラジェクトリの変換; remd\_convert)

conv1.inp:

```
[INPUT]
prmtopfile      = ../0_str/ww.prmtop
ambcrdfilere    = ../0_str/ww.crd
dcdfile         = ../3_reus/reus1_{}.dcd      # トラジェクトリーファイル(入力)
remfile         = ../3_reus/reus1_{}.rem      # レプリカの状態

[OUTPUT]
trjfile         = reus1_{}.dcd                # 出力するファイル名。{}を指定するのを忘れずに


[SELECTION]
group1          = ai:1-608                   # ここで指定した原子だけを書き出す
                                                    # 下のtrjout_atomで実際に指定
                                                    # 先頭から順番に取り出すのではなく、途中の原子だけを取り出すことも
                                                    # できるが、以後の解析で面倒なことが起こる可能性があるので
                                                    # できれば避けた方が良いかもしれない

[OPTION]
check_only      = NO
convert_type    = PARAMETER                  # パラメータでそろえる。REUSの場合は
                                                    # 同じアンブレラポテンシャルのデータまとめる
dcd_md_period   = 100                       # crdout_periodに相当
                                                    # remfileと合わせるために必須

trjout_format   = DCD
trjout_type     = COOR                       # 出力時に周期境界の設定を除去する
                                                    # 入力には周期境界有りで見られる
trjout_atom     = 1                         # 原子指定はここで行う
pbc_correct     = MOLECULE                   # 周期境界で分子がちぎれないようにする処理
```

# 実習(距離の計算; trj\_analysis)

自由エネルギー計算のために反応座標の値(重心間距離の和)を各アンブレラポテンシャルのデータについて計算する  
(今回の場合はエネルギーの出力から直接計算できなくもない)

 計算の実行(trj\_analysis)  
(各レプリカについて計算)

```
% cd ~/Tutorial1_2/5_trjana  
% sh trjana.sh &
```

 ログの確認

```
ログ確認(1番目)  
% less trjana1.log  
データ確認(1番目)  
% less dist1-1_1.dis  
% less distsum1-1_1.dis
```

dist1-1\_1.dis:

1	2.235	3.510
2	2.115	3.624
3	1.951	3.744
4	2.169	3.696
5	2.016	3.479
6	2.373	3.329
7	2.348	3.287
8	2.410	3.265
9	2.257	3.383
10	2.270	3.471
11	2.119	3.709
12	2.007	3.696
13	1.964	3.630
14	2.077	3.633
15	1.869	3.737
16	1.876	3.985
17	1.843	3.899

distsum1-1\_1.dis:

1	5.745
2	5.739
3	5.695
4	5.865
5	5.495
6	5.702
7	5.635
8	5.675
9	5.64
10	5.741
11	5.828
12	5.703
13	5.594
14	5.71
15	5.606
16	5.861
17	5.742

## 実習(距離の計算; trj\_analysis)

trjana1.inp: (通常のアンプレササンプリングの場合でも同じインプットを使えます)

```
[INPUT]
prmtopfile      = ../0_str/ww.prmtop          # 質量を使うタイプの計算では必須
ambcrdfile      = ../0_str/ww.crd             # ambcrdやpdbファイルだけでは代用できない

[OUTPUT]
comdisfile       = dist1-1_1.dis              # comdisfile は重心間距離出力用ファイル
                                                        # 他の値を計算する場合は違う名前を使う

[TRAJECTORY]
trjfile1         = ../conv/reus1_1.dcd
#trjfile2        = ../conv/reus2_1.dcd        # 二つ同時に扱う場合の例
md_step1         = 50000                     # REUS計算時のステップ数
mdout_period1    = 100                       # REUS計算時のcrdout_periodに合わせる
ana_period1      = 100                       # ステップ数の計算はmd_stepに準拠する
                                                        # mdout_period=ana_periodで全て計算するようになる
repeat1          = 1                         # 複数同時に処理する場合には1以外の数字
#repeat1         = 2                         # 二つ目のファイルがある場合
trj_format       = DCD
trj_type         = COOR                      # 変換前のトラジェクトリを使うならCOOR+BOX
trj_natom        = 608                      # トラジェクトリ中の原子数指定
                                                        # [INPUT]での入力から変えるために必要

[SELECTION]
(REUS計算の時と同じ。スペースが足りなかったので省略) # 今回は先頭から順番に取り出しているなのでこれでOK

[OPTION]
check_only       = NO                       # analysis_tool での共通パラメータ
com_distance1    = 1 2                      # SELECTION で指定したグループの
com_distance2    = 3 4                      # 重心距離を計算する(距離の和は計算できない)
```

# 実習(自由エネルギー計算; wham\_analysis)

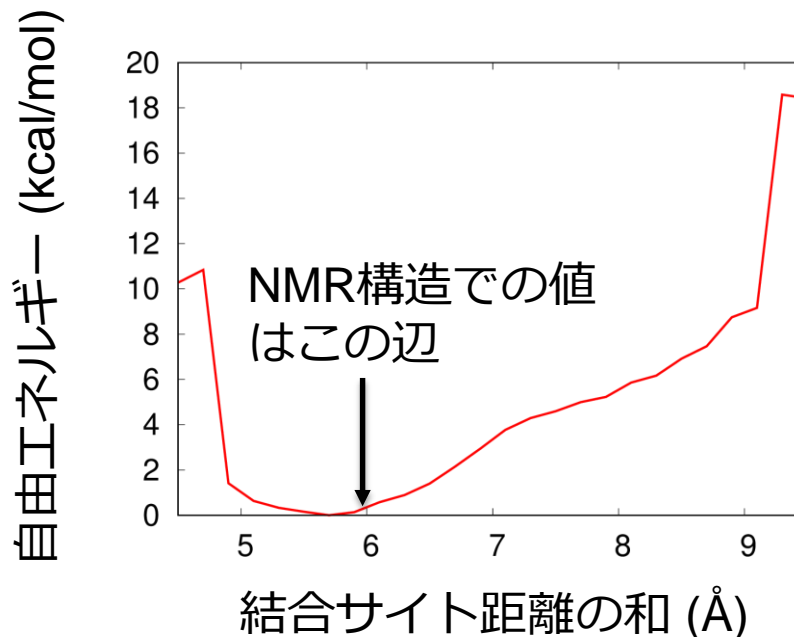
WHAM(Weighted Histogram Analysis Method)法で自由エネルギー計算

## ✎ 計算の実行(wham\_analysis)

```
% cd ~/Tutorial_2/6_wham  
% sh wham.sh &
```

## ✎ ログの確認

```
ログ確認  
% less wham.log  
自由エネルギーデータ  
% less output.pmf
```



(今回は計算が短すぎるので精度的には非常にアレです)

# 実習(自由エネルギー計算; wham\_analysis)

wham.inp:

```
[INPUT]
cvfile          = ../5_trjana/distsum1-1_{}.dis  # {} は REUS(MD)と同じ意味
                                                       # ここで与えた軸に沿った
                                                       # 自由エネルギープロファイルが
                                                       # 計算される


[OUTPUT]
pmffile         = output.pmf                    # 自由エネルギーの出力ファイル

[WHAM]
dimension       = 1
nblocks         = 1
temperature     = 300.0
tolerance       = 10E-08
rest_function1  = 1                             # どの拘束関数を使うかの指定
grids1          = 4.0 22.0 91                  # 自由エネルギープロファイルの
                                                       # グリッドの定義
                                                       # (最小値) (最大値) (グリッド数)

[RESTRAINTS]
constant1       = 1.0 1.0 1.0 1.0              # REUSの時と同じ値をいれる
reference1      = 6.0 7.0 8.0 9.0
```

(MBAR(Multistate Bennett Acceptance Ratio)法を用いた場合でも同様の計算は可能)

# まとめ

 GENESIS でレプリカ交換アンブレラサンプリング

 自由エネルギーを計算するまでの流れを簡単に実習



GENESIS ウェブサイト(※英語)  
日本語での質問は可能になっています(要ユーザ登録)  
<http://www.aics.riken.jp/labs/cbrt/>

もう少し簡単な系でのチュートリアル(※英語)  
<http://www.aics.riken.jp/labs/cbrt/tutorial/remd-tutorials/tutorial-2-2/>



# おまけ（もうすこし大きなデータを使った場合）

Tutorial\_2/x\_appendix 以下にもう少し大きな計算のデータがあります。  
これを使って自由エネルギー計算を試してみましょう。  
(使用した中間ファイル等もある程度置いてあるので参考にしてください)

✎ 距離範囲 6 Å ~ 21 Å まで(16レプリカ), レプリカ当たり 50 ns の計算

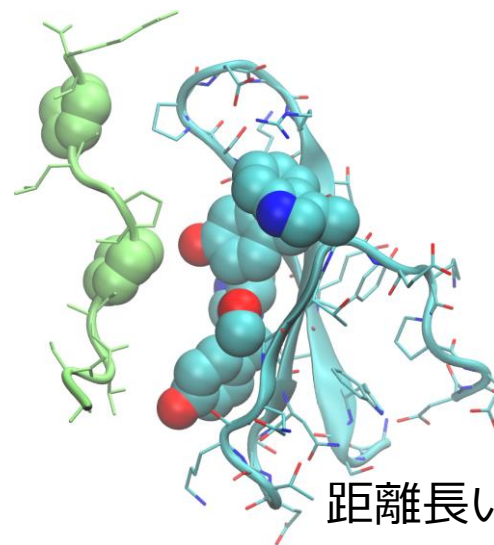
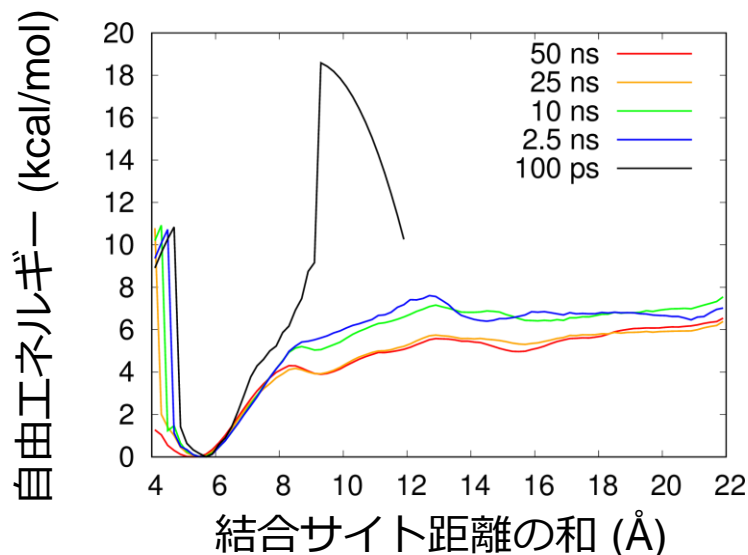
✎ 距離の和のファイル(Tutorial\_2/x\_appendix/trjana)

2.5 ns までのデータ: distsum1-5\_1.dis ... distsum1-5\_16.dis

10 ns までのデータ: distsum1-20\_1.dis ... distsum1-20\_16.dis

25 ns までのデータ: distsum1-50\_1.dis ... distsum1-50\_16.dis

50 ns 全体のデータ: distsum1-100\_1.dis ... distsum1-100\_16.dis



距離長いところの構造