最小自由エネルギー経路計算法

寺田　透

平成 26 年 7 月 22 日

1　自由エネルギー地形

自由度 N の系のカノニカルアンサンブルを考える。系の座標を $\boldsymbol{x} = (x_1, \ldots, x_N)$ とすると、確率密度関数は以下で与えられる。

$$\rho(\boldsymbol{x}) = Z^{-1} \exp\left[-\frac{V(\boldsymbol{x})}{k_{\mathrm{B}}T}\right] \tag{1}$$

ここで、$V(\boldsymbol{x})$ はポテンシャルエネルギー関数、k_{B} はボルツマン因子、T は温度、Z は

$$Z = \int \exp\left[-\frac{V(\boldsymbol{x})}{k_{\mathrm{B}}T}\right] d\boldsymbol{x} \tag{2}$$

である。N 次元の $\boldsymbol{x}$ から、n 次元の $\boldsymbol{z} = (z_1, \ldots, z_n)$ への変換 $\boldsymbol{z} = \boldsymbol{\theta}(\boldsymbol{x})$ を考える。

$$\boldsymbol{\theta}(\boldsymbol{x}) = (\theta_1(\boldsymbol{x}), \ldots, \theta_n(\boldsymbol{x})) \tag{3}$$

$\boldsymbol{z}$ 空間における、確率密度分布 $P(\boldsymbol{z})$ は、

$$P(\boldsymbol{z}) = \int \rho(\boldsymbol{x})\, \delta(\boldsymbol{z} - \boldsymbol{\theta}(\boldsymbol{x}))\, d\boldsymbol{x} \tag{4}$$

となる。従って、自由エネルギー地形 $F(\boldsymbol{z})$ は以下で与えられる。

$$\begin{aligned} F(\boldsymbol{z}) &= -k_{\mathrm{B}}T \ln P(\boldsymbol{z}) \\ &= -k_{\mathrm{B}}T \ln Z^{-1} \int \exp\left[-\frac{V(\boldsymbol{x})}{k_{\mathrm{B}}T}\right] \delta(\boldsymbol{z} - \boldsymbol{\theta}(\boldsymbol{x}))\, d\boldsymbol{x} \end{aligned} \tag{5}$$

2　最小自由エネルギー経路

状態 A から状態 B への遷移（立体構造変化）を考える。2 つの状態を繋ぐ経路のうち、実際の遷移で通過する確率が最も高い経路が最小自由エネルギー経路である。

最小自由エネルギー経路と類似した概念に、最小エネルギー経路がある。これは、遷移状態から出発して、ポテンシャルエネルギーの勾配の方向に進み、始状態と終状態に至る経路のことである。経路上の点に静止した状態から出発した微小時間 Δt の間の運動を考える。運動方程式は、

$$\begin{cases} \dot{x}_k &= v_k \\ \dot{v}_k &= -\dfrac{1}{m_k}\dfrac{\partial V(\boldsymbol{x})}{\partial x_k} \end{cases} \tag{6}$$

と書ける。ここで v_k は速度、m_k は質量である。これを解くと、移動度 $\Delta \boldsymbol{x}$ は、

$$\Delta x_k = -\frac{1}{m_k}\frac{\partial V(\boldsymbol{x})}{\partial x_k}\frac{\Delta t^2}{2} \tag{7}$$

で与えられる。従って、経路を $\hat{\boldsymbol{x}}(\alpha) = (\hat{x}_1(\alpha), \ldots, \hat{x}_N(\alpha))$ と表すと、この経路上では、経路とポテンシャルエネルギーの勾配を質量で割ったものが平行になる。

$$\frac{\partial \hat{x}_k(\alpha)}{\partial \alpha} \parallel \frac{1}{m_k}\frac{\partial V(\hat{\boldsymbol{x}}(\alpha))}{\partial x_k} \tag{8}$$

ここで、$\alpha\,(0 \leq \alpha \leq 1)$ は経路上の位置を表すパラメータであり、$\hat{\boldsymbol{x}}(0)$ は始状態、$\hat{\boldsymbol{x}}(1)$ は終状態を表す。このような運動を仮定して得られた経路は、固有反応座標（Intrinsic Reaction Coordinate、IRC）と呼ばれる [1]。

これを $\boldsymbol{z}$ 空間における経路 $\hat{\boldsymbol{z}}(\alpha)$ に変換する。

$$\frac{\partial \hat{z}_i(\alpha)}{\partial \alpha} = \frac{\partial \theta_i(\hat{\boldsymbol{x}}(\alpha))}{\partial \alpha} = \sum_{k=1}^{N}\frac{\partial \theta_i(\hat{\boldsymbol{x}}(\alpha))}{\partial x_k}\frac{\partial \hat{x}_k(\alpha)}{\partial \alpha} \tag{9}$$

であるから、式 (8) を代入すると、

$$\frac{\partial \hat{z}_i(\alpha)}{\partial \alpha} \parallel \sum_{k=1}^{N}\frac{\partial \theta_i(\hat{\boldsymbol{x}}(\alpha))}{\partial x_k}\frac{1}{m_k}\frac{\partial V(\hat{\boldsymbol{x}}(\alpha))}{\partial x_k} \tag{10}$$

となる。$\boldsymbol{z}$ 空間におけるポテンシャルエネルギー関数を $U(\boldsymbol{z})$ とする（ただし $\boldsymbol{z}$ 空間で $\boldsymbol{x}$ 空間がすべて記述できるものと仮定する）と、

$$\frac{\partial V(\hat{\boldsymbol{x}}(\alpha))}{\partial x_k} = \frac{\partial U(\hat{\boldsymbol{z}}(\alpha))}{\partial x_k} = \sum_{j=1}^{n}\frac{\partial U(\hat{\boldsymbol{z}}(\alpha))}{\partial z_j}\frac{\partial \theta_j(\hat{\boldsymbol{x}}(\alpha))}{\partial x_k} \tag{11}$$

であるから、式 (10) は、

$$\frac{\partial \hat{z}_i(\alpha)}{\partial \alpha} \parallel \sum_{j=1}^{n}\sum_{k=1}^{N}\frac{1}{m_k}\frac{\partial \theta_i(\hat{\boldsymbol{x}}(\alpha))}{\partial x_k}\frac{\partial \theta_j(\hat{\boldsymbol{x}}(\alpha))}{\partial x_k}\frac{\partial U(\hat{\boldsymbol{z}}(\alpha))}{\partial z_j} \tag{12}$$

と書ける。

これを一般的な粗視化空間 $\boldsymbol{z}$ に拡張するには、自由エネルギー地形 $F(\boldsymbol{z})$ を用いて、

$$\frac{\partial \hat{z}_i(\alpha)}{\partial \alpha} \parallel \sum_{j=1}^{n} M_{ij}(\hat{\boldsymbol{z}}(\alpha))\frac{\partial F(\hat{\boldsymbol{z}}(\alpha))}{\partial z_j} \tag{13}$$

のように書き換える [2]。この妥当性については**??**節で議論する。行列 $\boldsymbol{M}$ の要素は、

$$M_{ij}(\boldsymbol{z}) = P(\boldsymbol{z})^{-1}\int \sum_{k=1}^{N}\frac{1}{m_k}\frac{\partial \theta_i(\boldsymbol{x})}{\partial x_k}\frac{\partial \theta_j(\boldsymbol{x})}{\partial x_k}\rho(\boldsymbol{x})\,\delta(\boldsymbol{z} - \boldsymbol{\theta}(\boldsymbol{x}))\,d\boldsymbol{x} \tag{14}$$

で与えられる [2]。従って、最小自由エネルギー経路を得るためには、状態 A と状態 B を繋ぎ、経路上の至る所で式 (13) を満たす経路を求めれば良い。

3 String法

式 (13) を満たす経路 $\hat{\boldsymbol{z}}(\alpha)$ を求めるために、経路を最適化する方法を考える。このために、経路が時間に依存して変化すると考え、変化は $\boldsymbol{M}\partial F/\partial \boldsymbol{z}$ のうち、経路に垂直な成分に比例するものとする。

$$\gamma \frac{d\hat{\boldsymbol{z}}(\alpha,t)}{dt} = -\boldsymbol{M}\left(\hat{\boldsymbol{z}}(\alpha,t)\right)\frac{\partial F\left(\hat{\boldsymbol{z}}(\alpha,t)\right)}{\partial \boldsymbol{z}} + \lambda(\alpha,t)\frac{\partial \hat{\boldsymbol{z}}(\alpha,t)}{\partial \alpha} \tag{15}$$

ここで γ は摩擦係数であり、最適化の速度を決める。また、$\lambda(\alpha,t)$ は、経路に平行な成分を差し引くための関数である。経路が収束した段階では、$\dfrac{d\hat{\boldsymbol{z}}(\alpha,t)}{dt} = 0$ であるから、

$$\boldsymbol{M}\left(\hat{\boldsymbol{z}}(\alpha)\right)\frac{\partial F\left(\hat{\boldsymbol{z}}(\alpha)\right)}{\partial \boldsymbol{z}} = \lambda(\alpha)\frac{\partial \hat{\boldsymbol{z}}(\alpha)}{\partial \alpha} \tag{16}$$

となり、式 (13) が満たされていることがわかる。

$\boldsymbol{M}\left(\hat{\boldsymbol{z}}(\alpha,t)\right)$ を求めるためには、式 (14) より、$\hat{\boldsymbol{z}}(\alpha,t) = \boldsymbol{\theta}(\boldsymbol{x})$ を満たす $\boldsymbol{x}$ について、アンサンブル平均を計算すればよい。この条件を満たす $\boldsymbol{x}$ をサンプルするために、束縛付き定温分子動力学シミュレーションを行う。ポテンシャルエネルギーは、以下で与えられる。

$$U_{\kappa,\boldsymbol{z}}(\boldsymbol{x}) = V(\boldsymbol{x}) + \frac{\kappa}{2}\sum_{j=1}^{n}\left(\theta_j(\boldsymbol{x}) - z_j\right)^2 \tag{17}$$

ここで、κ は束縛の強さを表す定数である。このポテンシャルエネルギーの下で得られる確率密度関数は、

$$\rho_{\kappa,\boldsymbol{z}}(\boldsymbol{x}) = Z_{\kappa,\boldsymbol{z}}^{-1}\exp\left[-\frac{U_{\kappa,\boldsymbol{z}}(\boldsymbol{x})}{k_{\mathrm{B}}T}\right] \tag{18}$$

となる。ここで、

$$Z_{\kappa,\boldsymbol{z}} = \int \exp\left[-\frac{U_{\kappa,\boldsymbol{z}}(\boldsymbol{x})}{k_{\mathrm{B}}T}\right] d\boldsymbol{x} \tag{19}$$

である。デルタ関数は分散が 0 の正規分布で近似できることを利用すると、

$$\begin{aligned}\lim_{\kappa\to\infty}\rho_{\kappa,\boldsymbol{z}}(\boldsymbol{x}) &= \lim_{\kappa\to\infty} Z^{-1}\exp\left[-\frac{V(\boldsymbol{x})}{k_{\mathrm{B}}T}\right] Z Z_{\kappa,\boldsymbol{z}}^{-1}\exp\left[-\frac{\kappa}{2k_{\mathrm{B}}T}\sum_{j=1}^{n}\left(\theta_j(\boldsymbol{x}) - z_j\right)^2\right] \\ &= P(\boldsymbol{z})^{-1}\rho(\boldsymbol{x})\,\delta\left(\boldsymbol{z} - \boldsymbol{\theta}(\boldsymbol{x})\right)\end{aligned} \tag{20}$$

と書ける。従って十分大きい κ を用いてシミュレーションを行えば、

$$\begin{aligned}M_{ij}(\boldsymbol{z}) &\simeq \int\sum_{k=1}^{N}\frac{1}{m_k}\frac{\partial\theta_i(\boldsymbol{x})}{\partial x_k}\frac{\partial\theta_j(\boldsymbol{x})}{\partial x_k}\rho_{\kappa,\boldsymbol{z}}(\boldsymbol{x})\,d\boldsymbol{x} \\ &\simeq \frac{1}{N_s}\sum_{\tau=1}^{N_s}\sum_{k=1}^{N}\frac{1}{m_k}\frac{\partial\theta_i(\boldsymbol{x}(\tau))}{\partial x_k}\frac{\partial\theta_j(\boldsymbol{x}(\tau))}{\partial x_k}\end{aligned} \tag{21}$$

のように近似的に求めることができる。ここで、τ はシミュレーションにおける時刻、N_s はシミュレーションから得られたサンプル数である。

次に $\partial F/\partial \boldsymbol{z}$ の計算法を考える。束縛付きのシミュレーションの結果得られる自由エネルギーを $F_\kappa(\boldsymbol{z})$ とすると、

$$F_\kappa(\boldsymbol{z}) = -k_{\mathrm{B}}T \ln Z_{\kappa,\boldsymbol{z}} \tag{22}$$

であるから、

$$\begin{aligned}
\frac{\partial F_\kappa(\boldsymbol{z})}{\partial z_j} &= -k_{\mathrm{B}}T Z_{\kappa,\boldsymbol{z}}^{-1} \int \frac{\partial}{\partial z_j} \exp\left[-\frac{U_{\kappa,\boldsymbol{z}}(\boldsymbol{x})}{k_{\mathrm{B}}T}\right] d\boldsymbol{x} \\
&= \int \frac{\partial U_{\kappa,\boldsymbol{z}}(\boldsymbol{x})}{\partial z_j} Z_{\kappa,\boldsymbol{z}}^{-1} \exp\left[-\frac{U_{\kappa,\boldsymbol{z}}(\boldsymbol{x})}{k_{\mathrm{B}}T}\right] d\boldsymbol{x} \\
&= \int \kappa(\boldsymbol{z}_j - \theta_j(\boldsymbol{x}))\, \rho_{\kappa,\boldsymbol{z}}(\boldsymbol{x})\, d\boldsymbol{x}
\end{aligned} \tag{23}$$

を得る。一方、式 (22) は、κ が大きい時は、

$$\begin{aligned}
F_\kappa(\boldsymbol{z}) &= -k_{\mathrm{B}}T \ln \int \exp\left[-\frac{U_{\kappa,\boldsymbol{z}}(\boldsymbol{x})}{k_{\mathrm{B}}T}\right] d\boldsymbol{x} \\
&= -k_{\mathrm{B}}T \ln \int \exp\left[-\frac{V(\boldsymbol{x})}{k_{\mathrm{B}}T}\right] \\
&\quad \times \exp\left[-\frac{\kappa}{2k_{\mathrm{B}}T} \sum_{j=1}^{n} (\theta_j(\boldsymbol{x}) - z_j)^2\right] d\boldsymbol{x} \\
&\simeq -k_{\mathrm{B}}T \ln \left\{ Z^{-1} \int \exp\left[-\frac{V(\boldsymbol{x})}{k_{\mathrm{B}}T}\right] \right. \\
&\quad \left. \times Z \left(\frac{2\pi k_{\mathrm{B}}T}{\kappa}\right)^{\frac{n}{2}} \delta(\boldsymbol{z} - \boldsymbol{\theta}(\boldsymbol{x}))\, d\boldsymbol{x} \right\} \\
&= F(\boldsymbol{z}) - k_{\mathrm{B}}T \ln Z \left(\frac{2\pi k_{\mathrm{B}}T}{\kappa}\right)^{\frac{n}{2}}
\end{aligned} \tag{24}$$

と書くことができる。従って、

$$\begin{aligned}
\frac{\partial F(\boldsymbol{z})}{\partial z_j} &\simeq \frac{\partial F_\kappa(\boldsymbol{z})}{\partial z_j} \\
&= \int \kappa(\boldsymbol{z}_j - \theta_j(\boldsymbol{x}))\, \rho_{\kappa,\boldsymbol{z}}(\boldsymbol{x})\, d\boldsymbol{x} \\
&\simeq \frac{1}{N_s} \sum_{\tau=1}^{N_s} \kappa[z_j - \theta_j(\boldsymbol{x}(\tau))]
\end{aligned} \tag{25}$$

により計算することができる。

実際の経路の最適化計算は次のように行う。経路上の点を運動を計算するために、まず、経路の離散化を行う。R 個の点で離散化する場合、$\alpha_p = \dfrac{p}{R-1}$ として、$\hat{\boldsymbol{z}}_p = \hat{\boldsymbol{z}}(\alpha_p)$ とする。次に、Δt 秒後の各点の位置を、Euler 法などを用いて式 (15) を解くことによって求めた後、各点の間隔が等しくなるように再配置する。これを、経路が収束するまで繰り返すことによって、最適な経路を求める。この方法は、string 法 [2] と呼ばれる。

4 On-the-fly string 法

String 法では、経路最適化の各ステップで、$\boldsymbol{M}$ と $\partial F/\partial \boldsymbol{z}$ を求めるために、N_s 個のサンプルを得るシミュレーションが必要となる。この手続きは煩雑であり、時間もかかるため、より簡便な “on-the-fly” string 法が提案されている [3]。

4.1 方法

ここでは、式 (15) は、以下のように書き直される。

$$\gamma \frac{d\hat{\boldsymbol{z}}(\alpha,t)}{dt} = -\widetilde{\boldsymbol{M}}(\boldsymbol{x}(t))\,\kappa\left[\hat{\boldsymbol{z}}(\alpha,t) - \boldsymbol{\theta}(\boldsymbol{x}(t))\right] + \lambda(\alpha,t)\,\frac{\partial \hat{\boldsymbol{z}}(\alpha,t)}{\partial \alpha} \tag{26}$$

ここで、$\widetilde{\boldsymbol{M}}$ の成分は、式 (21) で $N_s = 1$ としたものである。

$$\widetilde{M}_{ij}(\boldsymbol{z}) = \sum_{k=1}^{N} \frac{1}{m_k} \frac{\partial \theta_i(\boldsymbol{x})}{\partial x_k} \frac{\partial \theta_j(\boldsymbol{x})}{\partial x_k} \tag{27}$$

また、式 (15) における、$\partial F/\partial \boldsymbol{z}$ の置き換えには、式 (25) で $N_s = 1$ としたものを用いている。ここで、$\boldsymbol{M}$、$\partial F/\partial \boldsymbol{z}$ は、本来別々にアンサンブル平均をとったものであることに注意する必要がある。$\langle X \rangle$ を X のアンサンブル平均と書くことにすると、一般に $\langle A \rangle \langle B \rangle$ と、$\langle AB \rangle$ は異なる。

この問題を解決するために、$\boldsymbol{x}$ とは独立に運動する変数 $\boldsymbol{y}$ を導入し、式 (26) を以下のように書き換える。

$$\gamma \frac{d\hat{\boldsymbol{z}}(\alpha,t)}{dt} = -\widetilde{\boldsymbol{M}}(\boldsymbol{x}(t))\,\kappa\left[\hat{\boldsymbol{z}}(\alpha,t) - \boldsymbol{\theta}(\boldsymbol{y}(t))\right] + \lambda(\alpha,t)\,\frac{\partial \hat{\boldsymbol{z}}(\alpha,t)}{\partial \alpha} \tag{28}$$

このようにすると、右辺第 1 項のアンサンブル平均は、

$$\begin{aligned}
&\int \widetilde{\boldsymbol{M}}(\boldsymbol{x})\,\kappa\left[\boldsymbol{z} - \boldsymbol{\theta}(\boldsymbol{y})\right]\rho_{\kappa,\boldsymbol{z}}(\boldsymbol{x})\,\rho_{\kappa,\boldsymbol{z}}(\boldsymbol{y})\,d\boldsymbol{x}d\boldsymbol{y} \\
&= \left\{\int \widetilde{\boldsymbol{M}}(\boldsymbol{x})\,\rho_{\kappa,\boldsymbol{z}}(\boldsymbol{x})\,d\boldsymbol{x}\right\}\left\{\int \kappa\left[\boldsymbol{z} - \boldsymbol{\theta}(\boldsymbol{y})\right]\rho_{\kappa,\boldsymbol{z}}(\boldsymbol{y})\,d\boldsymbol{y}\right\} \\
&= \boldsymbol{M}(\boldsymbol{z})\,\frac{\partial F(\boldsymbol{z})}{\partial \boldsymbol{z}}
\end{aligned} \tag{29}$$

となり、別々にアンサンブル平均をとったものの積と等しくなる。

この方法に基づく経路の最適化は、次のように行う。まず、前節と同様に、経路を R 個の点 $\hat{\boldsymbol{z}}_p\,(p = 1, \ldots, R)$ で離散化する。更に、束縛付きポテンシャルエネルギー関数

$$U_{\kappa,\hat{\boldsymbol{z}}_p}(\boldsymbol{x}) = V(\boldsymbol{x}) + \frac{\kappa}{2}\sum_{j=1}^{n}\left[\theta_j(\boldsymbol{x}) - \hat{z}_{pj}\right]^2 \tag{30}$$

で平衡化した系を 2 コピーずつ、計 $2R$ コピー用意する。次いで、コピーそれぞれについて、束縛付き分子動力学シミュレーションを行う。ここではまず、時刻 t における $\boldsymbol{x}(t)$、$\boldsymbol{y}(t)$、$\hat{\boldsymbol{z}}_p(t)$ から $\boldsymbol{x}(t+\Delta t)$、$\boldsymbol{y}(t+\Delta t)$ を求める。並行して、以下により経路上の点の位置を更新する。

$$\hat{\boldsymbol{z}}_p^*(t+\Delta t) = \hat{\boldsymbol{z}}_p(t) - \gamma^{-1}\widetilde{\boldsymbol{M}}(\boldsymbol{x}(t))\,\kappa\left[\hat{\boldsymbol{z}}_p(t) - \boldsymbol{\theta}(\boldsymbol{y}(t))\right]\Delta t \tag{31}$$

この段階では、まだ式 (26) の第 2 項を考慮していないので、$\hat{\boldsymbol{z}}_p\left(t+\Delta t\right)$ に * を付した。式 (26) の第 2 項は、経路に沿った移動をキャンセルするためのものである。従って、経路上に等間隔に点を配置している場合は、$\hat{\boldsymbol{z}}_p^*\left(t+\Delta t\right)$ が等間隔になるように再配置すれば良い。これらの操作を経路が収束するまで繰り返すことにより、最適な経路を求めることができる。

4.2 Alanine dipeptide の $\alpha \to \beta$ 構造転移の最小自由エネルギー経路

Alanine dipeptide (Ace-Ala-Nme) は水溶液中で、α ヘリックス様の構造と、polyproline II ヘリックス様の構造をとる。ここでは、両安定構造の間の構造転移の最小自由エネルギー経路を求める。反応座標として Ala の二面角 ϕ、ψ を用いることにすると、式 (27) より、$\widetilde{\boldsymbol{M}}$ の成分は、

$$\widetilde{\boldsymbol{M}} = \begin{bmatrix} \displaystyle\sum_{k=1}^{N} \frac{1}{m_k} \left(\frac{\partial \phi\left(\boldsymbol{x}\right)}{\partial x_k} \right)^2 & \displaystyle\sum_{k=1}^{N} \frac{1}{m_k} \frac{\partial \phi\left(\boldsymbol{x}\right)}{\partial x_k} \frac{\partial \psi\left(\boldsymbol{x}\right)}{\partial x_k} \\ \displaystyle\sum_{k=1}^{N} \frac{1}{m_k} \frac{\partial \psi\left(\boldsymbol{x}\right)}{\partial x_k} \frac{\partial \phi\left(\boldsymbol{x}\right)}{\partial x_k} & \displaystyle\sum_{k=1}^{N} \frac{1}{m_k} \left(\frac{\partial \psi\left(\boldsymbol{x}\right)}{\partial x_k} \right)^2 \end{bmatrix} \tag{32}$$

と書ける。本シミュレーションを行う、μ^2lib のアプリケーションプログラムの概要は以下の通り。

4.2.1 プログラムの概要

まず初めに、二面角を束縛する `DiheRest` クラスを作成する（説明に使用しない変数、関数は省略している）。

```
 1 class DiheRest : public InteractionBase {
 2 protected:
 3   int num;
 4   int *list[4];
 5   double *z;
 6   double *THETA;
 7   double *kappa;
 8   double **dTHETA;
 9 public:
10   int get_ene_num(void);
11   void setup(Param& p, Conf& c, Parallel& para,Molecule &m);
12   void force(double *x,double **th_f,double *ene,MoleculeBase &m,
13     DynamicsContext &context);
14 };
```

μ^2lib では、相互作用を計算するクラスは、`InteractionBase` クラスの派生クラスとして作成する。束縛する二面角の数を変数 `num` に格納する（ここでは `num = 2`）。`list` は 4×`num`

の 2 次元配列で、list[j][i] に i 番目の二面角の j 番目の原子のインデックスを格納する。z、THETA、kappa は大きさ num の配列で、それぞれ、そのコピーにおける $\hat{z}_p$ の座標、二面角の値、力の定数を格納する。dTHETA は num $\times 12$ の 2 次元配列で、dTHETA[i][3*j] に i 番目の二面角の j 番目の原子の x 座標に関する微分を、dTHETA[i][3*j+1] に i 番目の二面角の j 番目の原子の y 座標に関する微分を、dTHETA[i][3*j+2] に i 番目の二面角の j 番目の原子の z 座標に関する微分を格納する（list とはインデックスの順番が逆になっていることに注意)。force() 関数は以下の通り。

```
 1 void DiheRest::force(double *x,double **th_f,double *ene,
 2   MoleculeBase &m,DynamicsContext &context)
 3 {
 4   int i,i3;
 5   double d,fac;
 6   double *cosTHETA,*sinTHETA;
 7   double *f;
 8
 9   ene[0]=0.0;
10 // Only master process calculates force and energy.
11   if(master) {
12     f=th_f[0];
13
14     ALLOCATE_DOUBLE_ARRAY(cosTHETA,num);
15     ALLOCATE_DOUBLE_ARRAY(sinTHETA,num);
16     Dihe::calc_torsion(x,list[0],list[1],list[2],list[3],
17       0,num,dTHETA,cosTHETA,sinTHETA);
18     vdAcos(num,cosTHETA,THETA);
19     for(i=0;i<num;i++) {
20       if(sinTHETA[i] < 0.0) {
21         THETA[i]*=-1.0;
22       }
23     }
24     for(i=0;i<num;i++) {
25       d=THETA[i]-z[i];
26       d-=2.0*PI*floor(d/(2.0*PI)+0.5);
27       ene[0]+=0.5*kappa[i]*SQR(d);
28       fac=-kappa[i]*d;
29       i3=list[0][i]*3;
30       f[i3++]+=fac*dTHETA[i][0];
31       f[i3++]+=fac*dTHETA[i][1];
32       f[i3  ]+=fac*dTHETA[i][2];
33       /***** 中略 *****/
34       i3=list[3][i]*3;
```

```
35        f[i3++]+=fac*dTHETA[i][9];
36        f[i3++]+=fac*dTHETA[i][10];
37        f[i3  ]+=fac*dTHETA[i][11];
38      }
39      FREE_ARRAY(cosTHETA);
40      FREE_ARRAY(sinTHETA);
41    }
42 }
```

16 行目の Dihe::calc_torsion() は Dihe クラスのユーティリティ関数で、座標 x と原子リスト（list[0]∼list[3]）、担当領域（0、num）から、二面角の座標に関する微分 dTHETA と二面角のコサイン、サインの値（cosTHETA、sinTHETA）を計算する。

次に、$\hat{z}_p$ の時間発展を担う Tstring クラスを作成する。

```
 1 class Tstring {
 2 protected:
 3   int num;
 4   double *z;
 5   double **M;
 6   double gamma,dt;
 7 public:
 8   void setup(DiheRest& r,double g,Conf& c);
 9   void update(DiheRest& r,Param& p,Parallel& para);
10   void rearrange(DiheRest& r,Parallel& para,bool write_crd);
11   void write_rst(DiheRest& r,char *fname);
12 };
```

ここで、gamma は $\hat{z}_p$ の質量、dt は時間刻みである。setup()、write_rst() は、それぞれ、Tstring クラスオブジェクトの初期化と、シミュレーションを再開する際に使用する、Amber フォーマットの束縛設定ファイルの書き出しを行う関数である。式 (27)、(31) の計算を行う update() 関数は以下の通りである。

```
 1 void Tstring::update(DiheRest& r,Param& p,Parallel& para)
 2 {
 3   int i,j,k,l,m,n;
 4   int **list;
 5   double msum,fac,gamma_inv,dti;
 6   double *THETAy,*kappa,**dTHETA,*mass,mass_inv,*z_new;
 7   int size,rank;
 8   MPI_Comm comm;
 9   MPI_Status status;
10
11   if(!para.get_local_master()) {
12     return;
```

```
13  }
14
15  size      = para.get_global_size();
16  rank      = para.get_global_rank();
17  comm      = para.get_global_comm();
18
19  mass      = p.get_molecule_param().get_mass();
20  list      = r.get_list();
21  kappa     = r.get_kappa();
22  dTHETA    = r.get_dTHETA();
23  gamma_inv = 1.0/gamma;
24
25  if(rank % 2 == 0) {   /* x */
26    ALLOCATE_DOUBLE_ARRAY(THETAy,num);
27    MPI_Recv(THETAy,num,MPI_DOUBLE,rank+1,rank+1,comm,&status);
28    ALLOCATE_DOUBLE_ARRAY(z_new,num);
29  } else {              /* y */
30    THETAy    = r.get_THETA();
31    MPI_Send(THETAy,num,MPI_DOUBLE,rank-1,rank,comm);
32  }
33
34  if(rank % 2 == 0) {                   /* x */
35    for(i=0;i<num;i++) {
36      fac=0.0;
37      for(j=0;j<num;j++) {
38        msum=0.0;
39        for(k=0;k<4;k++) {
40          m=list[k][i];
41          n=-1;
42          for(l=0;l<4;l++) {
43            if(list[l][j] == m) {
44              n=l;
45            }
46          }
47          if(n >= 0) {
48            mass_inv=1.0/mass[m];
49            msum+=mass_inv*dTHETA[i][k*3  ]*dTHETA[j][n*3  ];
50            msum+=mass_inv*dTHETA[i][k*3+1]*dTHETA[j][n*3+1];
51            msum+=mass_inv*dTHETA[i][k*3+2]*dTHETA[j][n*3+2];
52          }
53        }
```

```
54          M[i][j]=msum;
55        }
56      }
57      for(i=0;i<num;i++) {
58        fac=0.0;
59        for(j=0;j<num;j++) {
60          fac+=M[i][j]*kappa[j]*(z[j]-THETAy[j]);
61        }
62        z_new[i]=z[i]-gamma_inv*fac*dt;
63      }
64
65      for(i=0;i<num;i++) {
66        z[i]=z_new[i];
67      }
68      FREE_ARRAY(z_new);
69      FREE_ARRAY(THETAy);
70    }
71 }
```

この関数は、第3引数で、プロセスをコピー（イメージの数 $\times 2$）ごとに分割した Parallel クラスオブジェクトを受け取る。この計算は、各コピーのローカルマスタープロセスのみが行う（11〜13 行目）。また、前節で述べた、独立に運動する 2 つのコピーのうち、偶数のグローバルランクを持つ $\boldsymbol{x}$ のコピーを担うプロセスが $\hat{\boldsymbol{z}}_p$ の時間発展の計算を行うため、奇数のグローバルランクを持つ $\boldsymbol{y}$ のコピーを担うプロセスは二面角のデータを、$\boldsymbol{x}$ のコピーを担うプロセスに送る（25〜32 行目）。また、$\boldsymbol{x}$ のコピーを担うプロセスは、35〜56 行目で、行列 $\widetilde{\boldsymbol{M}}$ の計算を行う。ここで、$\partial\phi\left(\boldsymbol{x}\right)/\partial x_k$ は k が Ace のカルボニル炭素、Ala のアミド窒素、Cα、カルボニル炭素の各原子を指している場合のみ値を持ち、$\partial\psi\left(\boldsymbol{x}\right)/\partial x_k$ は k が Ala のアミド窒素、Cα、カルボニル炭素、Nme のアミド窒素の各原子を指している場合のみ値を持つことに注意すると、積が値を持つのは、2 つの二面角が共有している Ala のアミド窒素、Cα、カルボニル炭素のみであることがわかる。従って、i 番目の二面角を構成する原子のリストを順に見ていき（39、40 行目）、その原子が j 番目の二面角を構成する原子のリストにも存在する（43 行目）時のみ、二面角の微分の積を計算すればよいことになる（47〜52 行目）。最後に、このようにして計算した $\widetilde{\boldsymbol{M}}$ と、$\boldsymbol{y}$ から $\hat{\boldsymbol{z}}_p^*\left(t+\Delta t\right)$ を計算する（57〜63 行目）。$\hat{\boldsymbol{z}}_p^*\left(t+\Delta t\right)$ を等間隔になるように再配置する rearange() 関数は以下の通り。

```
1 void Tstring::rearrange(DiheRest& r,Parallel& para,bool write_crd)
2 {
3   int i,j,q,total_steps,save_crd_freq;
4   double d,d2,d_inv,len;
5   double *L,*z_all,*z_new,**dz;
6   int size,rank,master;
7   MPI_Comm comm;
```

```
 8  double z_offset;
 9  double *THETA;
10
11  if(para.get_local_master()) {
12
13    size      = para.get_global_size();
14    rank      = para.get_global_rank();
15    master    = para.get_global_master();
16    comm      = para.get_global_comm();
17
18    if(master) {
19      ALLOCATE_DOUBLE_ARRAY(z_all,(size*num));
20      ALLOCATE_DOUBLE_ARRAY(z_new,(size*num));
21      ALLOCATE_DOUBLE_MATRIX(dz,size,num);
22      ALLOCATE_DOUBLE_ARRAY(L,size);
23    }
24    MPI_Gather(z,num,MPI_DOUBLE,z_all,num,MPI_DOUBLE,0,comm);
25    if(master) {
26      L[0]=0.0;
27      for(i=1;i<size;i++) {
28        d2=0.0;
29        for(j=0;j<num;j++) {
30          dz[i][j]=z_all[i*num+j]-z_all[(i-1)*num+j];
31          z_offset=2.0*PI*floor(dz[i][j]/(2.0*PI)+0.5);
32          dz[i][j]-=z_offset;
33          z_all[i*num+j]-=z_offset;
34          d2+=SQR(dz[i][j]);
35        }
36        d_inv=1.0/sqrt(d2);
37        for(j=0;j<num;j++) {
38          dz[i][j]*=d_inv;
39        }
40        L[i]=L[i-1]+d2*d_inv;
41      }
42      d=L[size-1]/(double)(size-1);
43      q=1;
44      for(i=1;i<size-1;i++) {
45        len=d*(double)i;
46        while(L[q] < len && q < size-1) q++;
47        for(j=0;j<num;j++) {
48          z_new[i*num+j]=z_all[(q-1)*num+j]+(len-L[q-1])*dz[q][j];
```

```
49          }
50        }
51        for(j=0;j<num;j++) {
52          z_new[j]=z_all[j];
53          z_new[(size-1)*num+j]=z_all[(size-1)*num+j];
54        }
55        for(i=0;i<size*num;i++) {
56          z_new[i]-=2.0*PI*floor((z_new[i]+PI)/(2.0*PI));
57        }
58      }
59      MPI_Scatter(z_new,num,MPI_DOUBLE,z,num,MPI_DOUBLE,0,comm);
60      if(master) {
61        FREE_ARRAY(z_all);
62        FREE_ARRAY(z_new);
63        FREE_MATRIX(dz,size);
64        FREE_ARRAY(L);
65      }
66    }
67
68    MPI_Bcast(z,num,MPI_DOUBLE,0,para.get_local_comm());
69    r.set_z(z);
70     /***** z の書き出し（省略） *****/
71  }
```

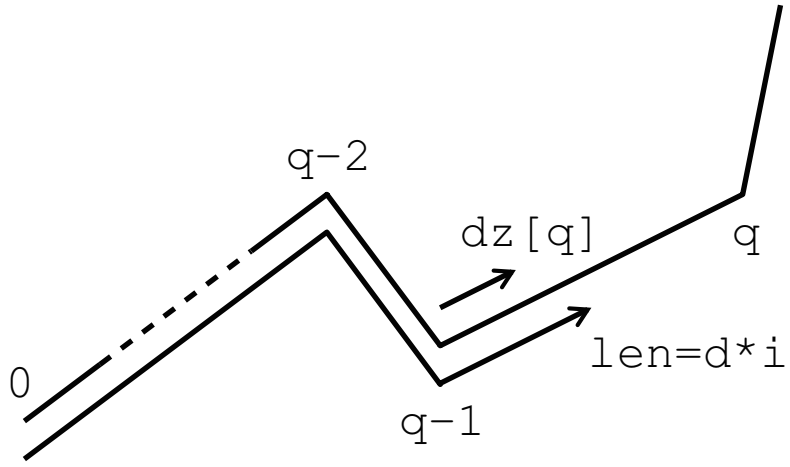

図 1: イメージの再配置

この関数は、第 2 引数で、プロセスをイメージごと（各イメージは $\boldsymbol{x}$、$\boldsymbol{y}$ の 2 つのコピーを含む）に分割した Parallel クラスのオブジェクトを受け取る（update() 関数とは異なるので注意）。$\hat{z}_p^*(t+\Delta t)$ の再配置は、グローバルマスターのみが行う（18 行目、25 行

目)。各イメージのマスターからzの内容をグローバルマスターのz_allに集めた後(24行目)、グローバルマスターはイメージ間の距離からstringの全長を求め(40行目)、新たなイメージの間隔を求める(42行目)。また、同時に隣接したイメージ間を結ぶ単位ベクトルdzを求める(38行目)。これらを用いて、図1のように、イメージの新たな座標を求める(43~54行目)。最後に、これを各イメージのマスターに分配し(59行目)、各イメージのマスターは、そのイメージに属するローカルプロセスにブロードキャストする(68行目)。各ローカルプロセスは、次のステップの計算に備えて新しいzをDiheRestクラスのオブジェクトにセットする(69行目)。main()関数は以下の通り。

```
 1 int main(int ac,char **av)
 2 {
 3   AmberParam p;
 4   AmberConf c;
 5   Molecule m;
 6   Dynamics a;
 7   Parallel para,para2;
 8   bool restart_flag;
 9   int i,ncopy,nsteps,thermostat,rearrange_freq;
10   double gamma;
11   char *rst_out_fname;
12   DiheRest r;
13   Tstring z;
14
15 // check args
16   ncopy=Conf::get_int_option(ac,av,"-ncopy");
17   gamma=Conf::get_double_option(ac,av,"-gamma");
18   rst_out_fname=Conf::get_string_option(ac,av,"-rst_out");
19   rearrange_freq=Conf::get_int_option(ac,av,"-rearrange_freq");
20
21 // start parallel
22   para.init(&ac,&av);
23   para2=para;
24   para.multicopy_mode(ncopy);
25   para2.multicopy_mode(ncopy/2);
26
27 /***** ファイルの読み込み(省略) *****/
28
29 // start simulation
30   srand48((long)c.get_random_seed());
31   c.set_restint(false);
32   m.add_interaction(&r);
33   a.setup(p,c,para,&m);
```

```
34   thermostat=c.get_thermostat();
35   nsteps=c.get_nsteps();
36   restart_flag=c.get_restart();
37   z.setup(r,gamma,c);
38   for(i=0;i<nsteps;i++) {
39     z.update(r,p,para);
40     if(thermostat == Conf::THERMOSTAT_TYPE_NONE) {
41       a.leapfrog(1,restart_flag);
42     } else if(thermostat == Conf::THERMOSTAT_TYPE_BERENDSEN) {
43       a.berendsen(1,restart_flag);
44     } else if(thermostat == Conf::THERMOSTAT_TYPE_LANGEVIN) {
45       a.langevin(1,restart_flag);
46     }
47     restart_flag=true;
48     if((i+1)%rearrange_freq == 0) {
49       z.rearrange(r,para2,((i+1)%c.get_save_crd_freq()==0));
50     }
51   }
52   if(para.get_local_master()) {
53     z.write_rst(r,rst_out_fname);
54   }
55   para.finalize();
56 }
```

ここでは、MoleculeBase クラスに内蔵されている RestInt クラスによって束縛の計算が行われないように設定し（31 行目）、代わりに、DiheRest クラスのオブジェクトを MoleculeBase クラスのオブジェクトに追加している（32 行目）。Tstring クラスのオブジェクトを初期化し（37 行目）、nsteps のシミュレーションを開始する（38 行目）。ここでは、まず時刻 t における $\boldsymbol{x}$ もしくは $\boldsymbol{y}$ の座標からその時点における自由エネルギーの勾配を求め、ここから $\hat{\boldsymbol{z}}_p^*(t+\Delta t)$ を求める（39 行目）。次いで、$\boldsymbol{x}$ もしくは $\boldsymbol{y}$ の座標を Δt 進める（40∼46 行目）。ここでは、DiheRest クラスのオブジェクトが保持しているイメージの座標 z は更新されていないので、時刻 t における束縛力が計算される。その後、$\hat{\boldsymbol{z}}_p^*(t+\Delta t)$ の再配置を行い、$\hat{\boldsymbol{z}}_p(t+\Delta t)$ を得る。更に、新しいイメージの座標を DiheRest クラスのオブジェクトに登録する（49 行目）。再配置は、$\boldsymbol{x}$ のコピーを担う全てのプロセスの間の通信を伴うため、毎ステップ再配置するのではなく rearrange_freq ステップごとに再配置するようにすることもできる。この間は、$\hat{\boldsymbol{z}}_p$ は更新されないが、$\hat{\boldsymbol{z}}_p^*$ は更新されていくので、結局、この期間の自由エネルギーの勾配の平均を用いて、$\hat{\boldsymbol{z}}_p$ を更新するのと同じことになる。シミュレーション終了後は、シミュレーションの再開に備え、現在の $\hat{\boldsymbol{z}}_p$ を Amber の束縛設定ファイル形式で書き出す（53 行目）。

4.2.2 シミュレーションの結果

まず、ϕ-ψ 空間上の初期パスに 8 つの点（イメージ）を配置した（$R = 8$）。各イメージの (ϕ, ψ) 座標は、それぞれ、$(-40.0, 130.0)$、$(-40.0, 105.0)$、$(-40.0, 80.0)$、$(-40.0, 55.0)$、$(-40.0, 30.0)$、$(-40.0, 5.0)$、$(-40.0, -20.0)$、$(-40.0, -45.0)$ とした。それぞれのイメージは独立に運動する 2 つのコピーを持つため、全体で 16 コピーのシミュレーションを行った。Alanine dipeptide の主鎖の二面角 (ϕ, ψ) を各イメージの座標付近に束縛した 300 K、1 bar の定温定圧分子動力学シミュレーションを行い、string 法のための初期構造とした。これらを用いて、300 K、1 bar の定温定圧条件下で string 法の計算を 1 ns 行った。ここでは、パラメータ κ は 1000 kcal mol^{-1} rad^{-2} とし、γ は 500 とした。再配置の頻度は 10 ステップごと（20 fs ごと）とした。Intel Xeon E5-2670 CPU を 2 基備える計算ノード 2 台（合計 32 コア）を用い、コピー当たり 1 プロセス 2 スレッドの条件でシミュレーションを行ったところ、1 ns のシミュレーションに約 1 時間 30 分かかった。図 2 は、初期パス（赤）、25 ps 後（緑）、50 ps 後（青）、100 ps 後（赤紫）、500 ps 後（水色）、1 ns 後（黄色）のパスを、別途求めた自由エネルギー地形の上に重ね合わせてプロットした図である。ここから、収束したパスは、polyproline II ヘリックス様構造の自由エネルギー極小状態から、α ヘリックス様構造の自由エネルギー極小状態をつなぐ、自由エネルギー障壁が最も低いパスに一致していることがわかる。図 3 の赤線は、

$$d(t) = \left[\frac{1}{R} \sum_{p=1}^{R} \left| \hat{\boldsymbol{z}}_p(t) - \hat{\boldsymbol{z}}_p(0) \right|^2 \right]^{\frac{1}{2}} \tag{33}$$

の時間変化を表している。ここから、0.4 ns 程度でパスが収束していることがわかる。緑線は、再配置の頻度を 1（毎ステップ）として同様なシミュレーションを行った結果を表しているが、収束の速度はほとんど変わらないことから、計算の効率を考慮すると、10 ステップに 1 回程度再配置すれば十分であることがわかる。図 4 において、赤線は後半 0.5 ns の $\hat{\boldsymbol{z}}_p$ の平均位置 $\overline{\boldsymbol{z}}_p$ における自由エネルギー勾配から計算した potential of mean force を表している。これは以下の式に従って計算した。

$$F_p = \frac{1}{2} \sum_{i=1}^{p} \sum_{\tau=1}^{N_s} \left[\overline{\boldsymbol{z}}_i - \boldsymbol{\theta}(\boldsymbol{y}(\tau)) \right] \cdot \left(\overline{\boldsymbol{z}}_{i+1} - \overline{\boldsymbol{z}}_{i-1} \right) \tag{34}$$

ここで、

$$\overline{\boldsymbol{z}}_0 = \overline{\boldsymbol{z}}_1 \tag{35}$$

$$\overline{\boldsymbol{z}}_{R+1} = \overline{\boldsymbol{z}}_R \tag{36}$$

である。このシミュレーションは 1 ns 行い、5 ステップ（10 fs）毎に保存した、$\boldsymbol{\theta}(\boldsymbol{y}(\tau))$ のトラジェクトリから、後半 0.9 ns 分を用いた（$N_s = 9 \times 10^4$）。また、緑線は、別途行った定温定圧シミュレーションにおいて、$\overline{\boldsymbol{z}}_p \pm 5^\circ$ の範囲の二面角をとる確率を求め、この値から自由エネルギーを求めたものである。String 法においては、離散化による誤差のためにピークの位置がずれているものの、自由エネルギー障壁の高さについては良く一致していることがわかる。

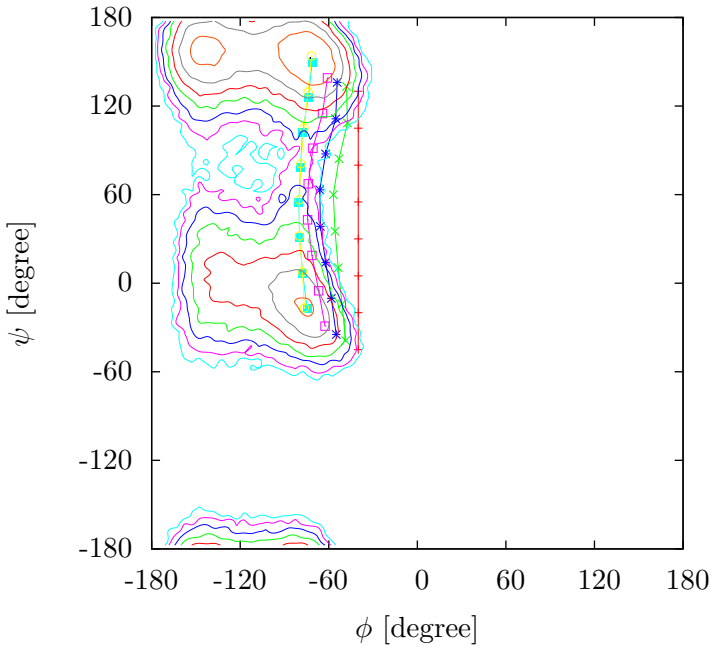

図 2: パスの時間変化

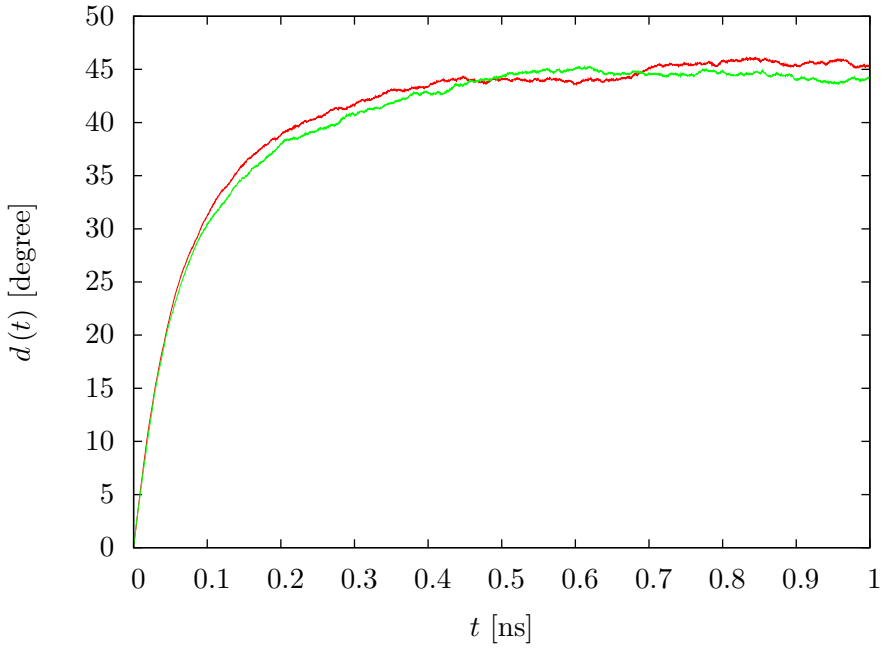

図 3: 初期パスからのずれの時間変化

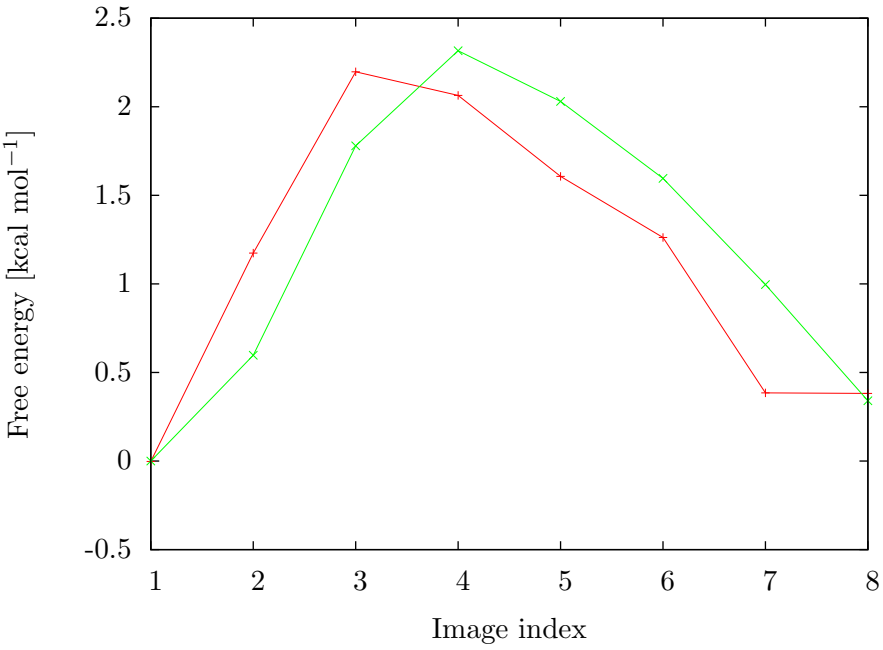

図 4: Potential of mean force の比較

参考文献

[1] Kenichi Fukui, Shigeki Kato, and Hiroshi Fujimoto "Constituent analysis of the potential gradient along a reaction coordinate. Method and an application to CH_4 + T reaction" *J. Am. Chem. Soc.* **97**, 1–7 (1975).

[2] Luca Maragliano, Alexander Fischer, Eric Vanden-Eijnden, and Giovanni Ciccotti "String method in collective variables: Minimum free energy paths and isocommittor surfaces" *J. Chem. Phys.* **125**, 024106 (2006).

[3] Luca Maragliano and Eric Vanden-Eijnden "On-the-fly string method for minimum free energy paths calculation" *Chem. Phys. Lett.* **446**, 182–190 (2007).