Frenkel法メモ

京都大学工学研究科材料工学科　西谷滋人

平成14年3月18日

概 要

Monte Carloシミュレーションを始める動機と，固体の自由エネルギーを精度よく求めるFrenkel法を用いて実際に計算した際のメモです.

目 次

1 Monte Carloシミュレーションを始める意義

何に使うか？
- Ti の高温の安定性への格子系の寄与を見積もる.
- 状態図計算をおこなう.

なぜ使うか？
- 定圧，定温のシミュレーションをする必要がある.
- force の計算があまりいらない.

どう使うか？
1. カノニカルアンサンブルで実働をテスト.
2. 並列化を検討.
3. isothermal-isobaric で Ti(EAM) の安定性を議論 (4 月締め切り).
4. 神藤さんの計算と合わす.
5. グランドカノニカルアンサンブルで TiAl(EAM) 状態図計算.
6. グランドカノニカルアンサンブルで BOP 状態図計算.

どう創るか？
1. potential.
2. MC のアルゴリズム.
3. NL 更新の効率.
4. Compressibility がなぜ必要か ?.

もともとの動機は Zr に対する Salomons の計算結果 [Salomons] がどの程度信頼できるかを調べるために始めた．まとめると，

> bcc-hcp エネルギーは EAM モデルを用いて 1850K 付近で逆転する結果が導かれる

である．しかし，

- 自由エネルギー差は 8meV 程度であり，それほどの精度が出るのか,
- 内部エネルギーの寄与がおおきい,
- ω相の安定性がわからない
- 系のおおきさの依存性がおおきい

等の問題があるように思われる．したがって，それらを検証することを第一の目標としている.

2 Monte Carloの原理

[Wm .G. Hoover, 計算統計力学, (森北出版，1999), pp.68–70.] 平衡統計力学の目標は，どんな観測できる量も，対応する微視的状態関数の平均として表すことである．正準集団では，状態の重みは $\exp(-E_i/kT)$ とわかっているので，この重みに比例する頻度で微視的状態を発生させるのはやさしい．そのような状態を生成し，重みつき平均を計算すれば，その平均は次第に正準集団平均

$$\langle \mathrm{Obs}(q,p) \rangle = \frac{\sum \mathrm{Obs}_i \exp(-E_i/kT)}{\sum \exp(-E_i/kT)} \tag{1}$$

に収束するだろう.

直接この平均をとる方法は，自由度が数個以上ある大抵の興味深い系では実用的でない．なぜなら，相空間のほとんどの部分が無視できる確率しか持たないからである．修正モンテカルロ標本化法はMetropolis, Resenbluth夫妻 Teller夫妻によって剛体円盤の状態方程式に応用された．[N.A.Metropolis, A.W. and M.N. Rosenbluth, A.H. andE. Tellr, J. Chem. Phys. 21, (1953),1087.] この方法は，配置空間における比較的小さなエネルギー変化 $\Delta E = kT$ と $\exp(-\Delta E/kT)$ に比例した状態遷移確率をもとにした，単純で使える方法，

$$\langle \mathrm{Obs} \rangle = \frac{\sum \mathrm{Obs}_i}{\sum 1} \tag{2}$$

であり，陽な重みがいらない．

モンテカルロ配置の列は，非対称な動きから生成される．ポテンシャルエネルギーを減らすような動きはすべて受け入れられる．ポテンシャルエネルギーを増やすような動きは，確率 $\exp(-\Delta E/kT)$ で受け入れられる．この方法は確率密度が最終的に $\exp(-E/kT)$ に収束することを保証する (以下で詳述).

3 Monte Carloのアルゴリズム

(N,V,T) が規定された正準集合 (canonical ensemble) を対象にした正準モンテカルロ法のアルゴリズムは以下の通り．

```
STEP_1 r_i=initial configuration
STEP_2 potential energy U_i
STEP_3 select one atom
STEP_4 r_j=try_move
STEP_5 potential eneryg U_j
STEP_6 if (dU=U_j-U_i<=0) {r_i=r_j;U_i=U_j;goto STEP_3;}
STEP_7 else R4=random
STEP-8 if (exp(-dU/kT) > R4)  {r_i=r_j;U_i=U_j;goto STEP_3;}
STEP-9 else {r_i;U_i;goto STEP_3;}
```

このアルゴリズムにしたがって作成したコードを以下に示す.

最大のステップ幅 d をどのように選択するかが常に問題となる．d が大きすぎると，試行ステップ数のわずかしか受け入れられないので確率密度関数 $p(x)$ のサンプリングが不十分ということになる．一方，d が小さすぎると，$p(x)$ のサンプリングが少なすぎるということになる． d のおおよその値は試行ステップ数の3分の1から2分の1といったところが妥当とされている．また，初期値は $p(x)$ ができるだけ速やかに非対称分布に近づくように設定する．これは $p(x)$ が最大値をとるような配置をとることで達成できる．

4 マルコフ過程と遷移確率

[Ueda p.90] サンプリングの手続きから明らかなように，ある時刻に状態 m を系がとるか否かは，その1ステップ前の系の状態 l にだけ依存し，それ以

List 3.1 NVT 一定 MC の Code のコア部分

```
int reject=0;
double Sum_U=0,TotalU;

double u_i=Energy.Total();      // STEP_2 u_i
TotalU=u_i;

for(int iter=1;iter<=iter_max;iter++){
    x=rand_generate(&ix);
    int select_atom=x*nbase;    // STEP_3 select atom

    try_move(select_atom);      // STEP_4 try_move

    double u_j=Energy.Total();  // STEP_5 u_j
    double dU=u_j-u_i;
    if( dU <= 0 ){              // STEP_6
        TotalU = u_i = u_j;
    } else {
        x=rand_generate(&ix);  // STEP_7
        if( exp(-dU/kT) > x){   // STEP_8
            TotalU = u_i = u_j;
        } else {    // STEP_9
            reset_move(select_atom); reject++;
        }
    }
    Sum_U += TotalU;
}
```

前に系がどんな状態を経てきたかにはよらない．このように系の状態が確率の法則に従い，次々遷移していく確率過程において，次のステップの状態が現在の状態だけで確率的に決まるとき，この確率過程をマルコフ過程 (Markoff process) と呼び，生成された状態の連鎖はマルコフの連鎖を構成するという．

前項で生成された状態の出現確率が，ステップ数 $n \to \infty$ のとき，カノニカル分布 $\exp(-U/k_{\mathrm{B}}T)$ に比例することは，以下のようにして示される．いま状態 l にある系が k ステップで状態 m に遷移する確率を

$$P_{lm}^{(k)}, P_{lm}^{(1)} \equiv P_{lm} \tag{3}$$

とおくと，マルコフ過程の場合，この遷移確率に対してチャップマン- コルモゴロフの式 (Chapman-Kolmogorov equation)

$$P_{lm}^{(k)} = \sum_j P_{lj}^{(k-1)} P_{jm} \tag{4}$$

が成立する．ここで和は $k-1$ ステップで系のとりうるすべての状態についてとる．マルコフ過程の遷移確率について次の定理がある．

もしすべての状態が同一のエルゴード状態に属するならば，

$$\lim_{k\to\infty} P_{lm}^{(k)} = w_m, m = 1, 2, \cdots, s \tag{5}$$

なる極限分布 w_m がすべての状態 m(s は状態の総数) に対して存在し，w_m は初期状態によらず，

$$w_m > 0, m = 1, 2, \cdots, s \tag{6}$$

である．また極限分布 w_m は，(4) 式で $k \to \infty$ とした方程式

$$w_m = \sum_j w_j P_{jm} \tag{7}$$

を満足する．

ここでエルゴード状態とは，エルゴード状態に属する任意の状態 l と m に対して，$P_{lm}^k > 0$ となる有限の k が存在する．つまり状態 l から m へ有限回の遷移で到達できるような状態の集合をさす．この定理の証明は [S. Karlin: A First Course in Stochastic Processes(Academic Press, 1966), 訳：佐藤健一, 佐藤由身子：確率過程講義 (産業図書，1974)] を参照されたい．

そこでカノニカル分布を実現するには，$w_m = \exp\left(-U_m/k_{\mathrm{B}}T\right)$ とおき，(7) 式を満足する遷移確率 P_{jm} を求めればよい．(7) 式を

$$\begin{aligned}\exp\left(-U_m/k_{\mathrm{B}}T\right) &= \sum_j \exp\left(-U_j/k_{\mathrm{B}}T\right) P_{jm} \\ &= \exp\left(-U_m/k_{\mathrm{B}}T\right) P_{mm} + \sum_{j\neq m} \exp\left(-U_j/k_{\mathrm{B}}T\right) P_{jm}\end{aligned}$$

と書き換えて，この左辺に

$$1 = \sum_j P_{mj} = P_{mm} + \sum_{j\neq m} P_{mj}$$

を掛けると

$$\sum_{j\neq m} \exp\left(-U_m/k_{\mathrm{B}}T\right) P_{mj} = \sum_{j\neq m} \exp\left(-U_j/k_{\mathrm{B}}T\right) P_{jm}$$

を得る．遷移確率 P_{mj} はこの式を満足するように選べばよい．そのためには

$$\exp\left(-U_m/k_{\mathrm{B}}T\right) P_{mj} = \exp\left(-U_j/k_{\mathrm{B}}T\right) P_{jm} \tag{8}$$

が成り立てばよい．これは十分条件である．j の代わりに l を用いて

$$P_{ml} : P_{lm} = 1 : \exp\left(-\left(U_m - U_l\right)/k_{\mathrm{B}}T\right)$$

と書き換えると，$U_m > U_l$ ならば，状態 m から l へは確率 1 で系を遷移させ，l から m へは確率 $\exp\left(-\left(U_m - U_l\right)/k_{\mathrm{B}}T\right)$ で遷移させれば良いことが分かる．さきに述べたサンプリングのアルゴリズムは，この結果に基づいている．

関係 (8) 式は，詳細釣り合いの原理 (principle of detailed balance) と呼ばれる関係であって，粒子のミクロな動きによる状態 m から l への遷移の起こる頻度が，逆の l から m への頻度と等しいことを示す．カノニカル分布で表された熱平衡状態はこの原理によって実現することが保証されるのである．

5　自由エネルギーの求め方

MC は平衡状態のシミュレーションを効率的にしてくれるが，その計算結果だけから entropy 等が直接求まるわけではない．参照状態から対象状態ま

での MC をおこない，その間を数値積分することによって値を求める必要がある．

必要となる積分のための関係は熱力学関数の微分形で求められる．カノニカル分布ではヘルムホルツの自由エネルギーは [Kubo, p.200]

$$F(T,V,N) = E - TS(E,V,N) \tag{9}$$

あるいは

$$-\frac{1}{T}F(T,V,N) = S(E,V,N) - \frac{E}{T} \tag{10}$$

で与えられる．ただし右辺の E は

$$\frac{\partial S(E,V,N)}{\partial E} = \frac{1}{T} \tag{11}$$

によって，V,T,N の関数とみなされる．以上の関係の組は $-F/T$ が S からの (独立変数を E から $1/T$ に変えた) Legendre 変換[1] であることを示す．したがって，

$$-\frac{\partial}{\partial\,(1/T)}\left(\frac{F}{T}\right) = \left(\frac{\partial S}{\partial T} - \frac{1}{T}\right)\frac{\partial E}{\partial\,(1/T)} - E = -E. \tag{12}$$

これより，エントロピー S は $1/T - E$ を，自由エネルギー F/T は $E - 1/T$ を積分することによって求まることが判る．自由エネルギーを求めるという操作は，MC シミュレーションでは $E - T$ を求めるということに帰着できる．

ある温度での参照状態の自由エネルギーが判っていれば，温度 T を変数として求めることが可能である．しかし，0K では発散してしまうのでたとえ微小温度を取ったとしても極端に精度が落ちる．一方，高温の自由ガスの状態を参照状態にしたとしても，固体の自由エネルギーを求めるには低温まで状態を移す間に相変態が起こり，信頼できる絶対値を得られない．そこで，Frenkel らが提唱している Einstein モデルからの遷移状態を使って求める方法が必要となる．[Bennett76]．この標準状態として Einstein model を用いる Frenkel らによって開発された方法 [**?**] について以下に詳しく見ていく．

基本的な考え方は以下の通りである．例えば，エントロピーを求める場合は，

$$S(E) = \int_{E_r}^{E} \frac{1}{T(E)} dE + S_r \tag{13}$$

[1] [Kubo, p.85] 一般に熱力学関数 (L) は，自然な独立変数を $x, y, z, \ldots$ としたとき，

$$dL = X dx + Y dy + Z dz + \ldots$$

という形の全微分表式 (Pfaff 形式) をもつ．ここに，$X, Y, Z, \ldots$ は $x, y, z, \ldots$ の関数である．独立変数と L の変換

$$\begin{aligned} L &\rightarrow \overline{L} = L - Xx \\ x, y, z, \cdots &\rightarrow X, y, z, \cdots \\ d\overline{L} &= -x\,dX + Y\,dy + Z\,dz + \cdots \end{aligned}$$

を一般に *Legendre* **変換**という．

である．ここで r は参照状態 (reference state) をさす．何らかのパラメータ Γ を導入すると以上の積分は

$$S(E) = \int_{E_r}^{E} \frac{1}{T(\Gamma)} \frac{dE(\Gamma)}{d\gamma} d\gamma + S_r \tag{14}$$

に変換される．

5.1 Einstein モデル

[Kittel, p.120 or Takahashi p.212] 振動数 ω の振動子の平均熱エネルギーは $\langle n \rangle \hbar\omega$ である．$\langle n \rangle$ はプランク分布 (Planck distribution)

$$\langle n \rangle = \frac{1}{1 - \exp(-\hbar\omega/\tau)} \tag{15}$$

$$\langle n \rangle + \frac{1}{2} = \frac{1}{2} \coth \frac{\hbar\omega}{2k_{\mathrm{B}}T} \tag{16}$$

で与えられる．

Einstein は相互作用しないがすべての原子が同一の共鳴振動数を持つと仮定したモデルを構築した．3 次元の N 個の振動子に対して，正準分配関数は,

$$Z_C = \left(\frac{1}{1 - \exp(-\hbar\omega/\tau)} \right)^{3N} \tag{17}$$

エネルギー U は

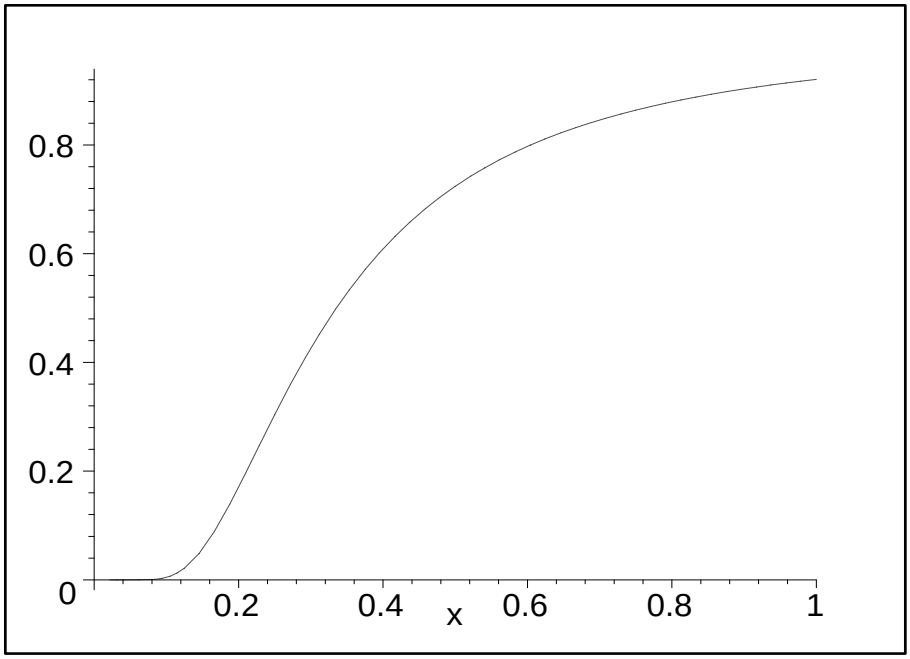
$$U = 3N \langle n \rangle \hbar\omega = \frac{3N\hbar\omega}{\exp(\hbar\omega/\tau) - 1} \tag{18}$$

である．ここで，$\tau = k_{\mathrm{B}}T$ である．比熱は

$$C_V = \left(\frac{\partial U}{\partial T} \right)_V = 3Nk_{\mathrm{B}} \left(\frac{\hbar\omega}{\tau} \right)^2 \frac{\exp(\hbar\omega/\tau)}{(\exp(\hbar\omega/\tau) - 1)^2} \tag{19}$$

となる．Maple による関数の外形を以下に示す.

```
> restart;
> U:=(x)->1/(exp(1/x)-1):
> d1:=unapply(diff(U(x),x),x):
> plot(d1(x),x=0..1);
```



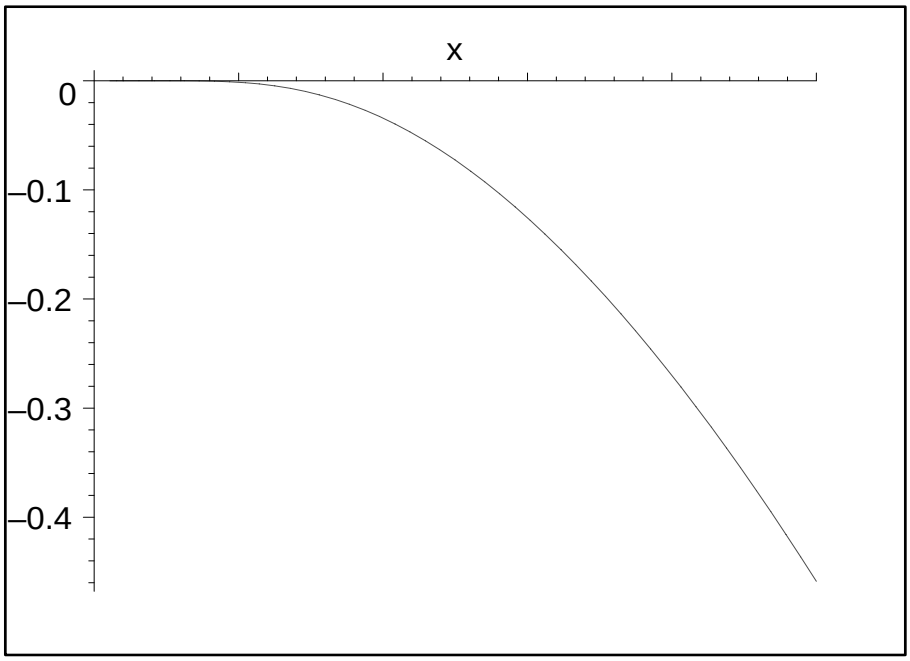

```
> int(U(1/x),x);
```

$$\ln(e^x - 1) - \ln(e^x)$$

```
> F:=x->ln(exp(x)-1)-ln(exp(x));
```

$$F := x \to \ln(e^x - 1) - \ln(e^x)$$

```
> plot(F(1/x)*x,x=0..1);
```



5.2 Frenkel 法

くわしい解説と導出は [Frenkel86a, p.171] に記されている．系の Hamiltonian を λ の関数 $H(\lambda)$ とする．free energy も λ に依存するため，

$$F(\lambda) = -kT \ln \left(\frac{1}{\Lambda^{3N} N!} \int \exp[-\beta U(\lambda)] d\mathbf{q}^N \right) \tag{20}$$

ポテンシャルエネルギーは今，λ にのみ依存すると限定する． kinetic energy の parameterizations は analytic に扱えるが，今はあまり興味がない．普通の手続きでは $U(\lambda)$ の形を $F(\lambda = 0)$ が分かっていて，$F(\lambda = 1)$ を求めたい free energy に取る．そうすると次の事実 (理解していない)

$$\frac{\partial F(\lambda)}{\partial \lambda} = \frac{\int \frac{\partial U(\lambda)}{\partial \lambda} \exp[-\beta U(\lambda)] d\mathbf{q}^N}{\int \exp[-\beta U(\lambda)] d\mathbf{q}^N} = \left\langle \frac{\partial U(\lambda)}{\partial \lambda} \right\rangle_\lambda \tag{21}$$

から $F(\lambda = 1)$ は

$$F(\lambda = 1) = F(\lambda = 0) + \int_0^1 \left\langle \frac{\partial U(\lambda')}{\partial \lambda'} \right\rangle_{\lambda'} d\lambda' \tag{22}$$

として求めることができる．ここで $\langle \partial U(\lambda)/\partial \lambda \rangle_\lambda$，つまり Hamiltonian $H(\lambda)$ で記述される系の機械的量 (mechanical quantity) $\partial U(\lambda)/\partial \lambda$ の canonical 平均は，コンピューターシミュレーション (MC や MD) によって計算される．

Einstein 法と呼ばれる Free energy の求め方では，以下のようなポテンシャルエネルギー関数を考える．

$$U(\lambda) = \lambda U(r^N) + (1 - \lambda) \left(K \sum_{i=1}^{N} (r^i - r_0^i)^2 + U_0(r_0^N) \right) \tag{23}$$

ここで U_0 はすべての原子がそれぞれのサイトにある場合のポテンシャルエネルギーである． $U(r)$ は原子間ポテンシャルを通じて相互作用をしている N 粒子の集合の和である． K はすべての原子を格子サイトに結びつけている可変のばね定数である．$\lambda = 1$ では，$U(\lambda)$ は結晶の相互作用を表し，$\lambda = 0$ では，$U(\lambda)$ は同じ構造の Einstein 結晶のポテンシャルエネルギー関数である．実際に K は Einstein 結晶内の原子の平均次乗変位が unperturbed 固体のそれと仮想的に一致するように選ばれる．

Einstein 結晶の自由エネルギーの絶対値は解析的に計算できるので，任意の同じ構造の自由エネルギーは熱力学的な積分で求められる．

$$\frac{\partial F}{\partial \lambda} = \left\langle V(r^N) - V_0 - K \sum_{i=1}^{N} (r^i - r_0^i)^2 \right\rangle \tag{24}$$

積分には 10-point の Gauss-Legendre 求積法か Bennett の overlapping-distribution 法が用いられる．

5.3 境界条件の改良

今までの計算では periodic boundary condition を用いてきた．前章で触れているように，Frenkel 法の λ =1 の条件でモンテカルロシミュレーションを

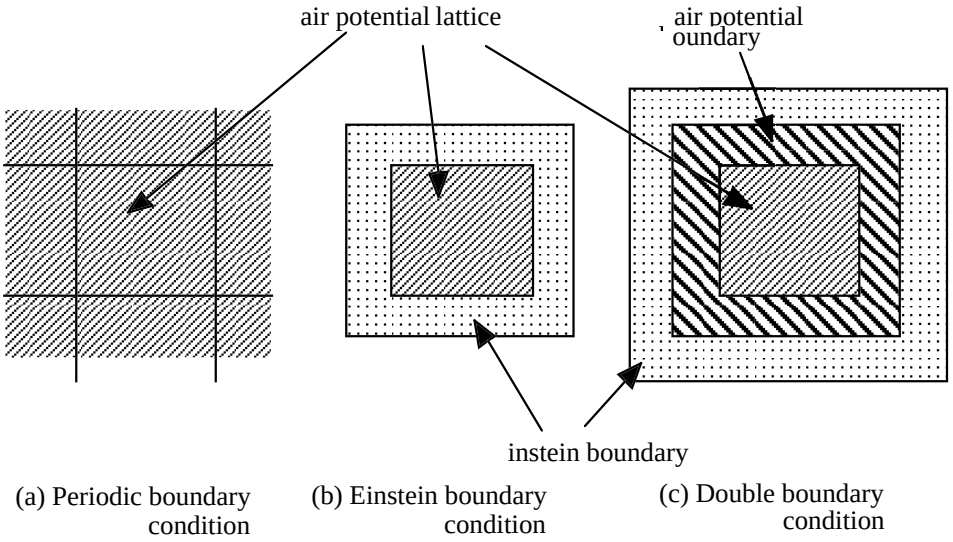


図 1: 種々の boundary condition の概念図

おこなうと，原子の集団移動（drift だけでなく rotation も）が起こってしまい，Einstein potential を適切に計算することができない．前章では Einstein potential のばね定数を大きくすることで，λ=1 に近い値まで，原子の集団移動を抑制して計算をおこなった．系のサイズを大きくすることでこの現象の発生をさらに抑えることは可能であるが，問題の本質的な解決にはならない．

ここでは boundary condition を periodic boundary condition から何らかのサイトに拘束された boundary condition へ変更したみた．この際，もっとも適切と思われるのがサイトの周りにばねで結びつけられる調和振動子 (Einstein 結晶) である．この条件を使うことによって，系は Einstein 結晶のなかに pair potential で拘束された集団が浮いているような描像となる．boundary の拘束の影響をさらに抑えるためには，ソフトバウンダリーのような二重セルのモデルを使い，十分に大きな境界の幅をとればよい．

これらのモデルの概念を図 1 に示す．Einstein 結晶を boundary に配置することによって drift を抑えることが可能である．適当な配置をモンテカルロシミュレーションにより生成すれば，$\left\langle \frac{\partial U(\lambda)}{\partial \lambda} \right\rangle$ を計算する中心部の原子の pair potential も適当な値が期待できる．より正確に求めるためには図 1(c) に示したように，$\left\langle \frac{\partial U(\lambda)}{\partial \lambda} \right\rangle$ を計算する中心部の結晶の周りに同じ pair potential を取る boundary 原子を配置し，そのさらに外側に Einstein boundary で囲むことによって，potential の違いによるモンテカルロシミュレーションで生成する配置の違いをさらに抑えることが可能と考えられる．この boundary condition を使うことによって，$\lambda = 1$ での原子の集団移動を抑えることも可能となり，積分計算の精度が格段に向上する．しかし，本研究では計算時間の関係から，図 1(b) で示した single boundary の結果を示す．

6 計算手順と結果

Frenkel 法により実際に自由エネルギーを求める手順をまとめておく.

1. NPT シミュレーションによる熱膨張（温度に対して sigma を求める〕
2. 一定の VT で，Λ を変えながら du/dl を求める
3. Lambda に対して du/dl を積分

6.1 mc の動かし方

`mc(return)`

で interactive な動作をする．入力例は以下の通り．これを `input` と打つと

List 6.1 interactive に mc に入力する場合

```
asura7/home/bob/MC# mc
Thu May 10 22:13:42 2001
Monte Carlo simulator developed by S.R.Nishitani with CodeWarrior.

Use default values?[Y or N]
If you want input format, type 'input':n
MC mode, 0:NVT, 1:NPT, 2:FRENKEL[1]? 1

Lattice 0:inter-, 1:on-site, 2:Eisntein[0]? 0

Potential, 0:Lennard-Jones, 1:EAM[0]? 0

Max iteration[200]?
```

⋮

以下略

`input.dat` にするデータ〔ひな形〕を出力してくれる．これを使って,

List 6.2 input.dat

```
no      //Use default Yes or No
1
1
0
200     //Max iteration
10      //Sampling Interval
100     //Average start iteration
600     //Temperature in Kelvin
0       //Pressure
0.01    //Atom move delta
0.01    //Volume adjust d
1.0     //Lambda for du/dl
0.7172472961 //0.7071067810 //sigma for volume
2   // Lattice size
0   // 0:FCC, 1:BCC
1.2 // R_cur for mkNeiborList
```

```
CommentStrip input.dat | mc(return)
```

としても走らす事が可能．相互作用 parameters は `Potential.parameters` から読み込む．

6.2 ポテンシャルパラメータ

ポテンシャルパラメータは `Potential.parameters` をホームディレクトリー (〜) にシンボリックリンクする．これは，rsh で他のマシンに振ったときに home directory で起動しているため．`Potential.parameters` は以下の通り．これは CommentStrip する必要はない．

List 6.3 Potential.parameters

```
//LJ
//      Einstein E_0 + K_const*r^2;
//K_const       E_0
//5.43  -2.1331647183
//5.43  -2.478314561
5.43    -2.4780

//      LJ r=sigma/dist
//      A * ( M * r^N - N * r^M);
//R_CUT2        A       M       N
1.8             0.355527452     5.5             9.0

//EAM
//      r=dist_0*norm(ij_vec)/sigma;
//      Einstein E_0 + K_const*r^2;
//K_const       E_0     dist_0=3.307*0.8660254;
//4.5           -2.424844794    2.863945998
4.5             -4.963364087881271      2.863945998

//      double r = sigma/dist;
//      cutting = -(r2/rc/r)^nc + (r1/rc)^nc;
//r1    r2      rc      nc
0.8660254       0.8660254000    1.4             7.5

//      double r = sigma/dist;
//      rep = AA * exp(-pp/r+pp) * exp (pp*cutting);
//      hop = BB * exp(-qq/r+qq) * exp (pp*cutting);
//R_CUT2        AA      BB      pp      qq
1.8             0.177621        2.020161        12.6    2.8
```

6.3 NPT シミュレーションによる熱膨張

温度に対して sigma を求める．`TempConrtol` に自動の shell script がある．これは，`input.dat2` にある基本設定を参照して，Tempearture を Values にある値に置き換えて，rsh で他のマシンで並列に計算させる．結果は `all.table` にある通り．第一近接の結果は `all1.table` の通り．元となる `input.dat2` のそれぞれの値の意味は mc を起動させたときに input と打てば説明されている．

List 6.4 TempControl

```
#!/bin/csh -f
set Values=(\
"10"\
"100"\
"200"\
"400"\
"500"\
"600"\
"800")
@ index=0

while ($index<7)
    @ index2 = $index + 1
    sed "s/#TEMP/$Values[$index2]/" input.dat2>input.tmp
    echo "#TEMPERATURE:" $Values[$index2] >asura$index.res
    CommentStrip input.tmp>input.tmp0
    rsh -l bob asura$index ~/MC/mc<~/MC/input.tmp0>>asura$index.res &
    echo asura$index
    @ index++
    @ time_count=0
    while ($time_count<600)
        @ time_count++
    end
end

wait

grep "#" asura*.res > all.res
awk -F: '{print $3}' all.res | formatter > all.table
```

List 6.5 all.table

```
asura7/home/bob/MC# cat  all.table
 10,  -2.477125182183769 ,   0.000000000000000 ,0.717147258728626,
 100,  -2.465690575472639 ,   0.000000000000000 ,0.716056577752542,
 200,  -2.452357993999382 ,   0.000000000000000 ,0.714870891114268,
 400,  -2.426005733886823 ,   0.000000000000000 ,0.712850615366218,
 500,  -2.414156661513016 ,   0.000000000000000 ,0.711557231257863,
 600,  -2.399422049141447 ,   0.000000000000000 ,0.710291050642509,
 800,  -2.373286283944128 ,   0.000000000000000 ,0.708009021872868,
```

List 6.6 all1.table

```
10, -2.131883687431261, 0.706997624306225,
100,-2.120846545495993, 0.705831470021174,
200,-2.107144582107388, 0.704502548187933,
400,-2.080422434016538, 0.701652715731670,
500,-2.065335644938758, 0.700378503280130,
600,-2.053725326418318, 0.698910740741895,
800,-2.023790068268947, 0.696082282455625,
```

List 6.7 input.dat2

```
no      //Use default Yes or No
1
1
0
3000     //Max iteration
10      //Sampling Interval
1000     //Average start iteration
#TEMP     //Temperature in Kelvin
0       //Pressure
0.01    //Atom move delta
0.01    //Volume adjust d
1.0     //Lambda for du/dl
0.707106781 //sigma for volume
2   // Lattice size
0   // 0:FCC, 1:BCC
1.2 // R_cur for mkNeiborList
```

6.4 du/dl の Lambda 依存

次に一定の VT で，LambdaControl は Lambda を変えながら du/dl を求める．All_calc は各温度での熱膨張から始めて，すべての温度での du/dl の Lambda 依存を計算する．

6.5 du/dl の積分

Lambda に対して du/dl を積分．生データを fit してもあまりきれいに fit できず，次数をあげると振動してしまう．そこで片 log でデータをフィットしてその関数を積分．Lambda=0 でプラスの値をとる場合があるので，原点をずらし，-0.1 程度にするとフィットがきれい．本来は，ばね定数が LJ と調和振動子で一致するように調整する必要があるが，ここでは積分を調整．FreeE.txt は

```
maple < FreeE.txt
cat FreeE.res >> final.table
```

として使える maple script.

```
> restart:with(stats):with(plots):with(linalg):
> data:=readdata("data1.res",4):

Warning, the name changecoords has been redefined

Warning, the protected names norm and trace have been redefined and
unprotected
> y0:=+1+data[1][3];
> data1:=map(u->[u[1],u[3]],data):
                        y0 := 1.010223906
> nn:=nops(data):
> for i from 1 to nn do
> data[i][3]:=data[i][3]-y0;
> od:
```

List 6.8 LambdaControl

```
#!/bin/csh -f
set Values=(\
"0.0"\
"0.1"\
"0.4"\
"0.7"\
"0.8"\
"0.9"\
"0.99")
@ index=0

while ($index<7)
    @ index2 = $index + 1
    sed "s/#LAMBDA/$Values[$index2]/" input.dat.lambda>input.tmp
    echo "#LAMBDA:" $Values[$index2] >asura$index.res
    CommentStrip input.tmp>input.tmp0
    rsh -l bob asura$index ~/MC/mc<~/MC/input.tmp0>>asura$index.res &
    echo asura$index
    @ index++
    @ time_count=0
    while ($time_count<600)
        @ time_count++
    end
end

wait

grep "#" asura*.res > all.res
awk -F: '{print $3}' all.res | formatter > all.table
```

List 6.9 input.dat.all

```
no      //Use default Yes or No
#MODES
#FINISH     //Max iteration
#INTERVAL      //Sampling Interval
#START     //Average start iteration
#TEMPERATURE     //Temperature in Kelvin
0       //Pressure
0.01    //Atom move delta
0.01    //Volume adjust d
#LAMBDA     //Lambda for du/dl
#SIGMA //sigma for volume
2   // Lattice size
0   // 0:FCC, 1:BCC
1.2 // R_cur for mkNeiborList
```

List 6.10 All_calc

```
#!/bin/csh -f
#
#

# for WARIKOMI p.234
set Values=(\
"0.0"\
"0.1"\
"0.4"\
"0.7"\
"0.8"\
"0.9"\
"1.00")

onintr catch
if ($#argv<2) then
        echo "Usage: $0 start_step finish_step (sampling interval=10)"
        exit 1
endif

set start=$1
set finish=$2
if ($#argv>2) then
    set interval=$3
else
    set interval=10
endif

sed -f MODE_NPT input.dat.all>tmp
sed "s/#FINISH/$finish/\
s/#INTERVAL/$interval/\
s/#START/$start/\
s/#LAMBDA/1.0/\
s/#SIGMA/0.707106781/\
s/#TEMPERATURE/#TEMP/" tmp > input.dat2


./TempControl
echo "Subject: [calc] Whole loop " $1 "-" $2 " with " $3 " step" > final.table
date >> final.table
cat all.table >> final.table
```

List 6.10 All_calc(continued)

```
sed -f MODE_FRENKEL input.dat.all>tmp
sed "s/#FINISH/$finish/\
s/#INTERVAL/$interval/\
s/#START/$start/" tmp > tmp1

set list="all.table"

@ index_temp=1
while (${index_temp}<8)
    set sigma=‘sed -n "${index_temp}p" all.table|awk -F, ’{print $4}’‘
    set temp=‘sed -n "${index_temp}p" all.table|awk -F, ’{print $1}’‘
    echo $temp $sigma
    sed "s/#TEMPERATURE/$temp/" tmp1>tmp2
    sed "s/#SIGMA/$sigma/" tmp2>input.dat.lambda
    @ index_temp++
    @ index_lambda=0

    while ($index_lambda<7)
        @ index_lambda2 = $index_lambda + 1
        sed "s/#LAMBDA/$Values[$index_lambda2]/" input.dat.lambda>input.tmp
        echo "#LAMBDA:" $Values[$index_lambda2] >asura$index_lambda.res
        CommentStrip input.tmp>input.tmp0
        rsh -l bob asura$index_lambda ~/MC/mc<~/MC/input.tmp0>>asura$index_lambd
a.res &
        echo asura$index_lambda
        @ index_lambda++

        @ time_count=0
        while ($time_count<1200)
            @ time_count++
        end

    end

    wait
    while ($time_count<1200)
        @ time_count++
    end

    grep "#" asura*.res > all.res
    echo "Temperature:"$temp ", Sigma:" $sigma >> final.table
    awk -F: ’{print $3}’ all.res | formatter >> final.table
    awk -F: ’{print $3}’ all.res | formatter | sed ’s/,/ /g’> data1.tmp
    maple < FreeE.txt
    cat FreeE.res >> final.table
    date >> final.table
end
exit 0

catch:
        onintr -
        echo "Interruped on mailing \"
        echo $name
        exit 1
```

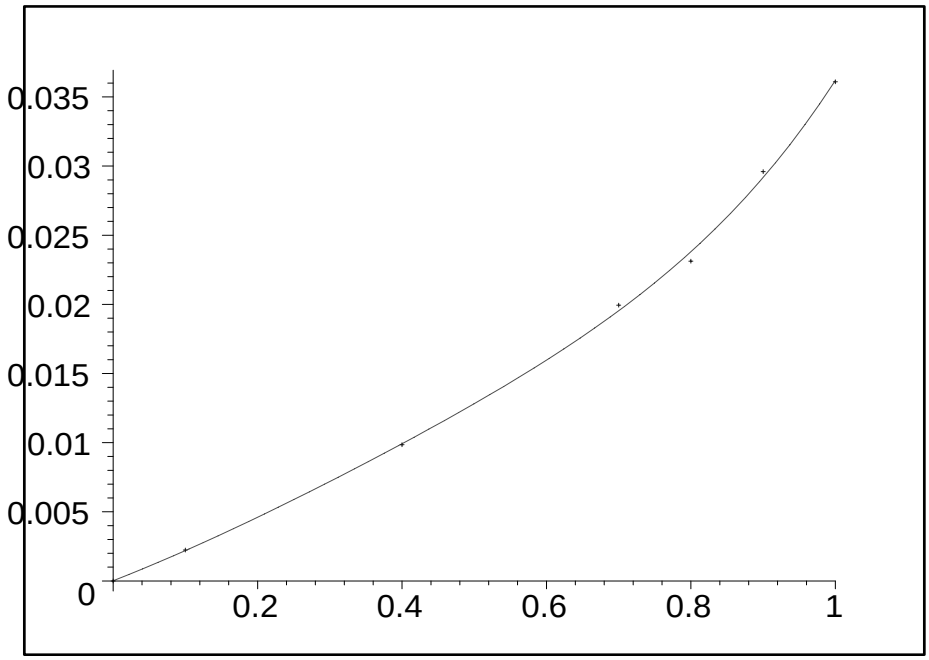
List 6.11 MODE_NPT

```
/#MODES/ c\
1 \
1 \
0
```

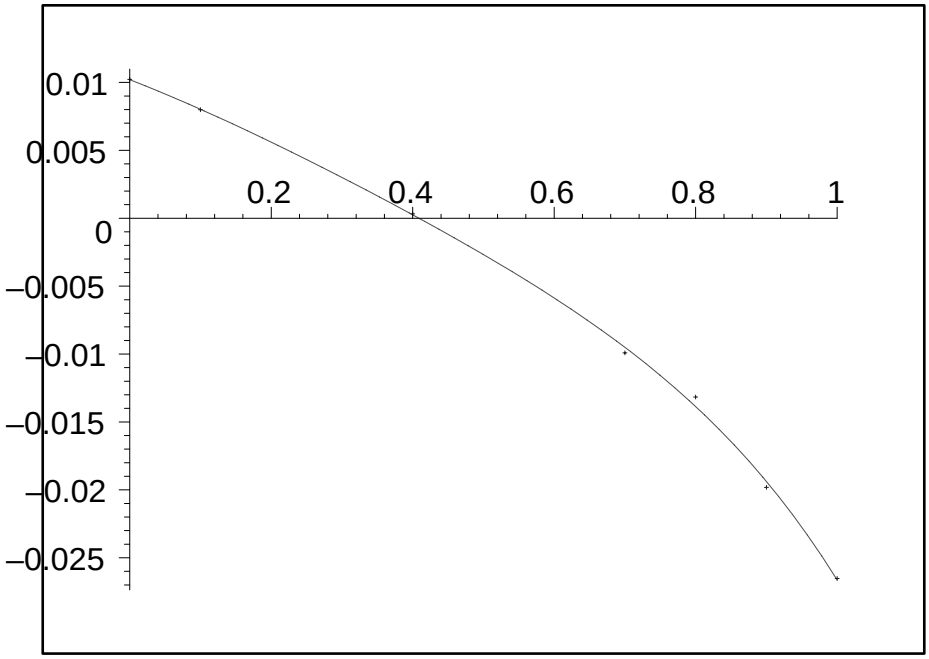
List 6.12 MODE_FRENKEL

```
/#MODES/ c\
2 \
2 \
0
```

```
> data2:=map(u->[u[1],log(-u[3])],data):
> data3:=convert(transpose(convert(data2,array)),listlist):
> fit2:=fit[leastsquare[[x,y], y=c0+c1*x+c2*x^2+c3*x^3+c4*x^4+c5*x^5,
> {c0,c1,c2,c3,c4,c5}]](data3):
> f2:=unapply(rhs(fit2),x):
> p_1:=plot(data2,color=blue,style=POINT):
> p_f1:=plot(f2(x),x=0.0..1.0):
> display(p_1,p_f1);
```




```
> p_1:=plot(data1,color=blue,style=POINT):
> p_f1:=plot(-exp(f2(x))+y0,x=0.0..1.0,color=red):
> display(p_1,p_f1);
```




```
> evalf(int(-exp(f2(x))+y0,x=0..1));
```

$$-.004209809713$$

List 6.13 FreeE.txt

```
interface(quiet=true);
with(stats):with(linalg):
data:=readdata("data1.tmp",4):
data1:=map(u->[u[1],u[3]],data):
nn:=nops(data):
y0:=0.1+data[1][3]:
 for i from 1 to nn do
     data[i][3]:=data[i][3]-y0:
 od:
data2:=map(u->[u[1],log(-u[3])],data):
data3:=convert(transpose(convert(data2,array)),listlist):
fit2:=fit[leastsquare[[x,y], y=c0+c1*x+c2*x^2+c3*x^3+c4*x^4+c5*c^5, {c0,c1,c2,c3
,c4,c5}]](data3):
f2:=unapply(rhs(fit2),x):
result:=evalf(int(-exp(f2(x))+y0,x=0..1)):
writeto("FreeE.res"):
printf("Integrated Free Energy: %15.10f\n",result):
writeto(terminal):
interface(quiet=false):
```

6.6 最終結果

第 2 近接モデルでの結果 (`final.table`).

List 6.14 final.table

```
Subject: [calc] Whole loop  1000 - 3000  with  10  step
Tue May  8 15:31:57 JST 2001
 10,  -2.477125182183769 ,   0.000000000000000 ,0.717147258728626,
 100,  -2.465690575472639 ,   0.000000000000000 ,0.716056577752542,
 200,  -2.452357993999382 ,   0.000000000000000 ,0.714870891114268,
 400,  -2.426005733886823 ,   0.000000000000000 ,0.712850615366218,
 500,  -2.414156661513016 ,   0.000000000000000 ,0.711557231257863,
 600,  -2.399422049141447 ,   0.000000000000000 ,0.710291050642509,
 800,  -2.373286283944128 ,   0.000000000000000 ,0.708009021872868,
Temperature:10 , Sigma: 0.717147258728626
 0.0,  -2.476701623880100 ,  -0.000122886878287 ,0.717147258728626,
 0.1,  -2.476720986190981 ,  -0.000472997395306 ,0.717147258728626,
 0.4,  -2.476778250327854 ,  -0.001291754731917 ,0.717147258728626,
 0.7,  -2.476882046602075 ,  -0.002144085687763 ,0.717147258728626,
 0.8,  -2.476937013338801 ,  -0.002491942995084 ,0.717147258728626,
 0.9,  -2.476998368607434 ,  -0.002757535109488 ,0.717147258728626,
 1.00,  -2.477038153128471 ,  -0.003118735991222 ,0.717147258728626,
Integrated Free Energy:    -.0015999012
Tue May  8 15:55:30 JST 2001
Temperature:100 , Sigma: 0.716056577752542
 0.0,  -2.465020783396086 ,   0.002196702395315 ,0.716056577752542,
 0.1,  -2.465272518352583 ,   0.001631433762620 ,0.716056577752542,
 0.4,  -2.465172879565767 ,  -0.000580146149604 ,0.716056577752542,
 0.7,  -2.465202503524778 ,  -0.003165060215822 ,0.716056577752542,
 0.8,  -2.465323841928730 ,  -0.004612194320478 ,0.716056577752542,
 0.9,  -2.465219650675531 ,  -0.005728138705347 ,0.716056577752542,
 1.00,  -2.465379529370471 ,  -0.008630049803430 ,0.716056577752542,
Integrated Free Energy:    -.0018030357
Tue May  8 16:19:07 JST 2001
Temperature:200 , Sigma: 0.714870891114268
 0.0,  -2.452256801892137 ,   0.004868463548284 ,0.714870891114268,
 0.1,  -2.452311169949720 ,   0.003440828673914 ,0.714870891114268,
 0.4,  -2.452112201734125 ,  -0.000431509346245 ,0.714870891114268,
 0.7,  -2.452285650679872 ,  -0.004800091857096 ,0.714870891114268,
 0.8,  -2.452282314546932 ,  -0.007035293879867 ,0.714870891114268,
 0.9,  -2.452106837752670 ,  -0.011470899722679 ,0.714870891114268,
 1.00,  -2.452535987557672 ,  -0.014354358677830 ,0.714870891114268,
Integrated Free Energy:    -.0026330645
Tue May  8 16:42:50 JST 2001
Temperature:400 , Sigma: 0.712850615366218
 0.0,  -2.426369381144002 ,   0.010223905702848 ,0.712850615366218,
 0.1,  -2.426711870253616 ,   0.007987694388592 ,0.712850615366218,
 0.4,  -2.426601228812481 ,   0.000323941166345 ,0.712850615366218,
 0.7,  -2.426080581459298 ,  -0.009917573728701 ,0.712850615366218,
 0.8,  -2.426357735505321 ,  -0.013165754021106 ,0.712850615366218,
 0.9,  -2.426483859273380 ,  -0.019817539919035 ,0.712850615366218,
 1.00,  -2.425898432463118 ,  -0.026527886937421 ,0.712850615366218,
Integrated Free Energy:    -.0042098097
Tue May  8 17:06:40 JST 2001
```

List 6.14 final.table(continued)

```
Temperature:500 , Sigma: 0.711557231257863
 0.0,  -2.413260214044746 ,   0.012365732825709 ,0.711557231257863,
 0.1,  -2.413995105962358 ,   0.009061782279548 ,0.711557231257863,
 0.4,  -2.413635417097108 ,   0.000546519197662 ,0.711557231257863,
 0.7,  -2.412564716473067 ,  -0.014023912426861 ,0.711557231257863,
 0.8,  -2.414003747668451 ,  -0.016305736517589 ,0.711557231257863,
 0.9,  -2.413307182659751 ,  -0.021950945742296 ,0.711557231257863,
 1.00,  -2.413127209028266 ,  -0.033422314574031 ,0.711557231257863,
Integrated Free Energy:    -.0054804125
Tue May  8 17:30:19 JST 2001
Temperature:600 , Sigma: 0.710291050642509
 0.0,  -2.401062343150187 ,   0.014280756513975 ,0.710291050642509,
 0.1,  -2.400928096510216 ,   0.010972145150600 ,0.710291050642509,
 0.4,  -2.400407552777581 ,  -0.000562270122532 ,0.710291050642509,
 0.7,  -2.400369935572356 ,  -0.014268969275385 ,0.710291050642509,
 0.8,  -2.399955553625667 ,  -0.022026332361379 ,0.710291050642509,
 0.9,  -2.400994817491678 ,  -0.028199627211893 ,0.710291050642509,
 1.00,  -2.400221856442155 ,  -0.050763889656300 ,0.710291050642509,
Integrated Free Energy:    -.0074673399
Tue May  8 17:54:04 JST 2001
Temperature:800 , Sigma: 0.708009021872868
 0.0,  -2.373772442322081 ,   0.019110573465948 ,0.708009021872868,
 0.1,  -2.374423260444426 ,   0.014947792481552 ,0.708009021872868,
 0.4,  -2.374655047081285 ,  -0.002250677002858 ,0.708009021872868,
 0.7,  -2.375215606030175 ,  -0.019147495800667 ,0.708009021872868,
 0.8,  -2.373858276444851 ,  -0.033666054870812 ,0.708009021872868,
 0.9,  -2.374074142484935 ,  -0.045489139855569 ,0.708009021872868,
 1.00,  -2.373570137950830 ,  -0.064315275727469 ,0.708009021872868,
Integrated Free Energy:    -.0114062586
Tue May  8 18:17:48 JST 2001
```

6.7 プログラムのチェック

プログラム開発時にすべてのモードでの値をチェックするために使っていた Chcek，その他付属の shell scripts ChangeInput，CommentStrip，all_kill です.

List 6.15 Check

```
#!/bin/csh -f
set Values=(\
"1 0 0 1.0"\
"1 1 0 1.0"\
"1 1 1 1.0"\
"2 0 0 0.5"\
"2 1 0 0.5"\
"2 2 0 0.5"\
"2 2 1 0.5")

echo ‘date‘ > test.res
while ($#Values)
    ChangeInput $Values[1] > input.tmp
    echo "#" $Values[1] >> test.res
    ./CommentStrip input.tmp | mc >> test.res
    shift Values
end

grep "#" test.res > all.res
```

List 6.16 ChangeInput

```
#!/bin/csh -f

sed "s/#MC/$argv[1]/;s/#BOUND/$argv[2]/;s/#POTEN/$argv[3]/;s/#LAMBDA/$argv[4]/"
input.dat >input.tmp1

cat input.tmp1
```

7 moment 法と Frenkel 法の結果の比較

同じ LJ ポテンシャルを用いて，moment 法と Frenkel 法の結果の比較をおこなった.

7.1 Potential

Moment 法で採用されている LJ ポテンシャルの表式は

$$\psi(r) = \frac{D}{m-n}\left[n\left(\frac{r_0}{r}\right)^m - m\left(\frac{r_0}{r}\right)^n\right] \tag{25}$$

神藤らが使っている単位系は，erg(Kelvin)-m. $U_0 = -28760.49148$K となるが，神藤先生のコメントによると，

List 6.17 CommentStrip

```
#!/usr/bin/perl
# Chop "//" type comments from files
# by S.R.Nishitani 27/2/01

foreach $current_file (@ARGV){
  open(INPUT, "$current_file") || die "can't open $current_file $!\n";
  $delim="//";
#  $delim="#";
  while (<INPUT>){
#    s/\s//g;
    $num=index($_, $delim);
    if ($num<0){
      print $_, if $_ ne "";
#      print $_,"\n" if $_ ne "";
    } elsif ($num > 0){
      $outline=substr($_,0,$num);
      print $outline,"\n";
    }
  }
  print "\n";
  close (INPUT);
}
```

List 6.18 all_kill

```
#!/bin/csh -f
if ($#argv<1) then
        echo "Usage: $0 killing_process"
        exit 1
endif

ps -edf |grep $1 |awk '{print $2}' > /tmp/AD1$$
set line=`wc /tmp/AD1$$|awk '{print $1}'`
set ii=1
while ( $ii < $line)
      set number=`sed -n "${ii}p" /tmp/AD1$$`
      echo $number
      kill -9 $number
      @ ii++
end

rm /tmp/AD1$$

exit 0
```

```
so far U0=Kelvin unit, while psai0,psai=erg unit (99/8/15)
U0_eV=U0*8.617385e-05;
psai0_eV=psai0*6.2415064e+11;
U0_eV/psai0_eV = boltz_con:=1.380658e-16;
```

でeV へ直せる．U0 は-2.478eV となる．実際にMaple でみると，

```
> restart:
> psi:=r->DD/(m-n)*(n*(r0/r)^m-m*(r0/r)^n):
> m:=9:
> n:=5.5:
> r0:=2.5487e-08:
> DD:=4125.70:
> erg:=1.60219e-12:#1eV=1.60219e-12 erg
> Bolt_const:=1.38061e-16: #erg
> 1/erg;

                                        12
                              .6241457006 10

> r_eq:=fsolve(diff(12*psi(r)+6*psi(r*sqrt(2)),r)=0,r=r0);
> evalf(r0/r_eq/sqrt(2));

                                                -7
                           r_eq := .2512666221 10
                                  .7172472961

> U:=r->evalf(12*psi(r)+6*psi(r*sqrt(2)))/2:
> U(r_eq)/erg*Bolt_const;
> U(r0)/erg*Bolt_const;
> DD/erg*Bolt_const;

                                  -2.478296715
                                  -2.466683578
                                  .3555123099
```

mc プログラムのなかでパラメータとして使われている σ はこれから分かるように $(r_0/r_{eq})/\sqrt{2}$ に対応している．また，DD はeV に直すため，.3555 にする．

次に振動数 ω は，ばね定数 k，原子の質量 m を使って

$$\omega = \sqrt{\frac{k}{m}} \tag{26}$$

となる．単位は $1/s$ であるので，k を $\mathrm{eV/\AA^2}$ から SI 単位系 $\mathrm{J/m^2 = kg/s^2}$ に直す必要がある．この単位換算を含めると ω は

```
>Abogadoro:=6.02217e-23:
```

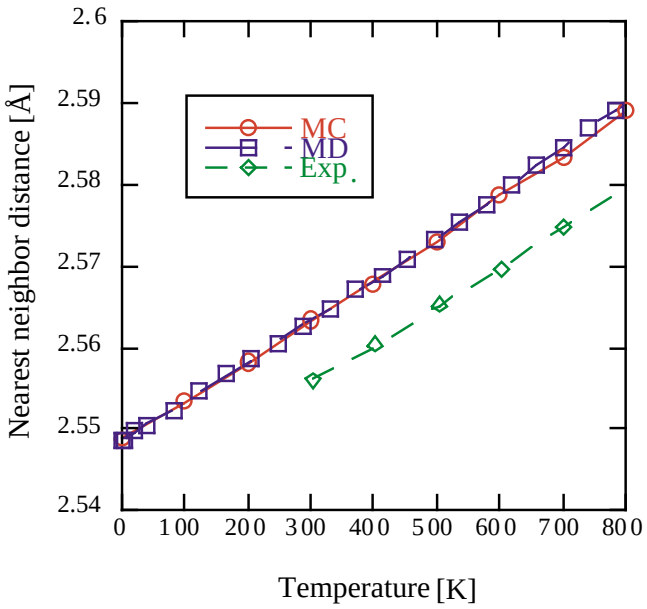


図 2: MC と MD による最近接原子間距離のシミュレーション結果の比較

```
> k_prime:=k*erg*10^(-7)/(10^(-10))^2:#eV/A^2->J/m^2
> M_prime:=M*10^(-3)/Abogadoro:# Relative atomic mass -> real atom mass
> omega:=sqrt(k_prime/M_prime);
                                              -9
                    omega := .9822759566 10    sqrt(k/M)
```

となる.

7.2 MD との比較

第一近接モデルで MD の結果との比較．最近接原子間距離，エネルギーはともに良い一致を示している．自由エネルギーの差は古典的なシミュレーションと半古典的なシミュレーションの違い．Frenkel 法で求めた自由エネルギーは Einstein model の量子的な結果から導かれていることに注意.

> 下野氏注：統計力学の教科書によると、調和振動子は、古典的には、ハミルトニアン $H = \frac{p^2}{2m} + \frac{m\omega^2 q^2}{2}$ に対し、古典統計による分配関数は $Z(\omega) = \frac{kT}{\hbar\omega}$、また、量子論的には、ハミルトニアンの固有値 $E_n = (n + 1/2)\hbar\omega$ に対し、量子統計による分配関数は $Z(\omega) = \left[2\sinh\frac{\hbar\omega}{2kT}\right]^{-1}$ となると思います。これらは、両方とも運動エネルギーの分を含んでいます。
>
> 上式より、自由エネルギーの温度依存性は、古典的には、0 からスタートし、温度が $\hbar\omega$ の程度までちょっと増えて、後は単調減少、量子論的には、ゼロ点振動のエネルギー $\frac{\hbar\omega}{2}$ から、単調減少、となります。

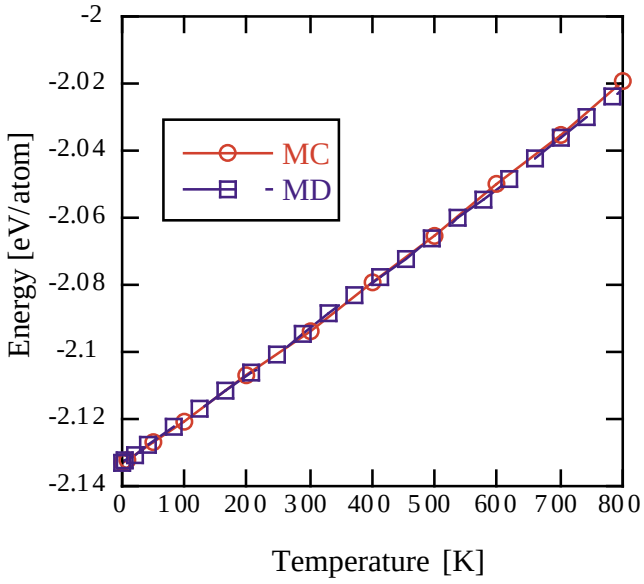


図 3: MC と MD による内部エネルギーのシミュレーション結果の比較

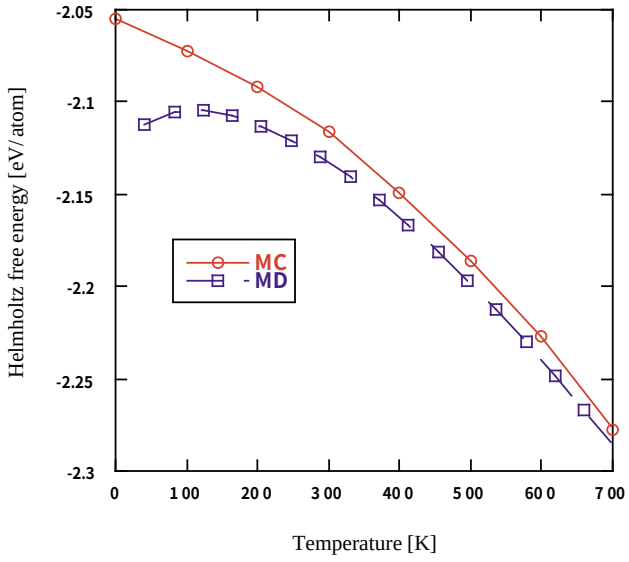


図 4: MC と MD による自由エネルギーのシミュレーション結果の比較

7.3 Moment 法との比較

第二近接モデルでの Moment 法との比較.

List 7.1 Moment.res

#T(K)	r1	a1	U0	harmonic	free
10.00	2.51357	3.55472	-0.28760E+05	0.49962E-13	0.16447E-15
100.00	2.51723	3.55989	-0.28758E+05	0.45410E-13	0.21289E-15
200.00	2.52162	3.56611	-0.28752E+05	0.18229E-13	0.49112E-15
300.00	2.52645	3.57294	-0.28740E+05	-0.28883E-13	0.98706E-15
400.00	2.53175	3.58043	-0.28721E+05	-0.90727E-13	0.17149E-14
500.00	2.53766	3.58879	-0.28694E+05	-0.16445E-12	0.27004E-14
600.00	2.54438	3.59830	-0.28655E+05	-0.24849E-12	0.39815E-14
700.00	2.55231	3.60951	-0.28599E+05	-0.34219E-12	0.56172E-14
800.00	2.56223	3.62353	-0.28513E+05	-0.44591E-12	0.77164E-14
900.00	2.57644	3.64364	-0.28364E+05	-0.56246E-12	0.10556E-13

List 7.2 mc.res

#temp	Internal E	du/dl	sigma	Integrated du/dl
10	-2.477125182183769	0.000000000000000	0.717147258728626	-.0015999012
100	-2.465690575472639	0.000000000000000	0.716056577752542	-.0018030357
200	-2.452357993999382	0.000000000000000	0.714870891114268	-.0026330645
400	-2.426005733886823	0.000000000000000	0.712850615366218	-.0042098097
500	-2.414156661513016	0.000000000000000	0.711557231257863	-.0054804125
600	-2.399422049141447	0.000000000000000	0.710291050642509	-.0074673399
800	-2.373286283944128	0.000000000000000	0.708009021872868	-.0114062586

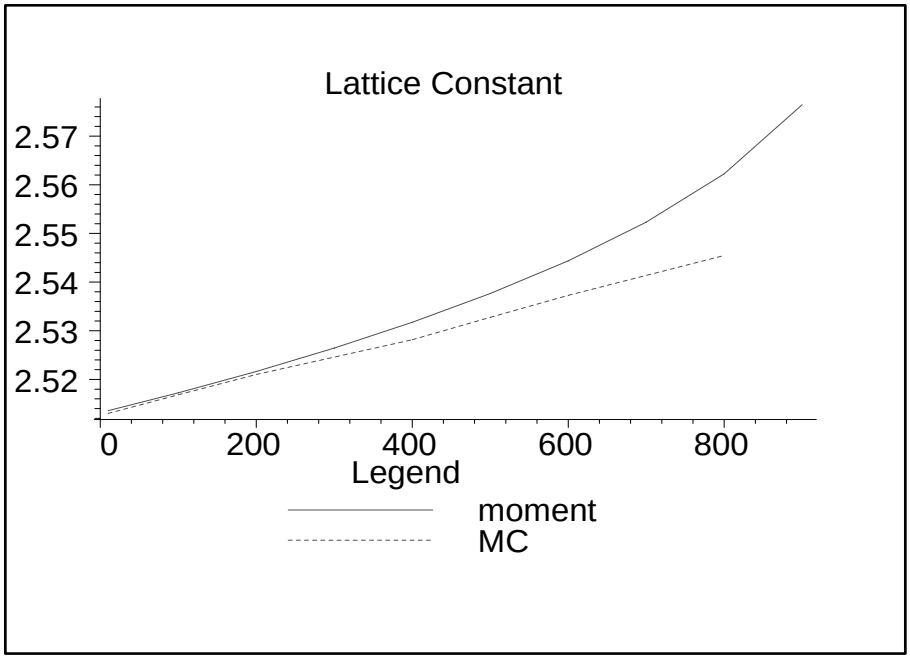
```
> restart:with(plots):

Warning, the name changecoords has been redefined
> Avogadro:=6.02217e+23:
> ergtoeV:=1.60219e-12:
> ergtoJ:=1.0e7:
> k_B:=1.38061e-16:  #erg
> k_B:=k_B/ergtoeV;
> hbar:=6.62620e-27:  #erg
> hbar:=evalf(hbar/ergtoeV/2/Pi);
```

$$k\_B := .00008617017957$$

$$hbar := .6582193643\,10^{-15}$$

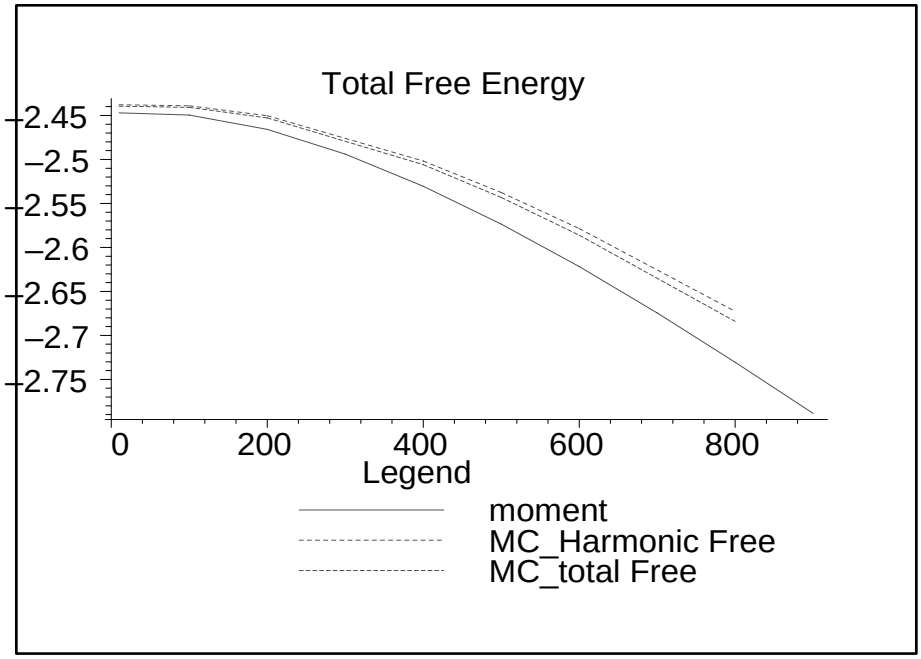
```
> A:=readdata("moment.res",6):
> A0:=map(u->[u[1],u[2]],A):#Lattice Const [A]
> A1:=map(u->[u[1],u[4]*k_B],A):#Internal Energy [
> A2:=map(u->[u[1],u[4]*k_B+u[5]/ergtoeV],A):#Harmonic Free Energy
> A3:=map(u->[u[1],u[4]*k_B+(u[5]+u[6])/ergtoeV],A):#Free Energy
> B:=readdata("mc.res",5):
> r_eq:=2.5487:
> B0:=map(u->[u[1],evalf(r_eq/u[4]/sqrt(2))],B):
> p1:=plot(A0,linestyle=1,legend="moment"):
> p2:=plot(B0,linestyle=2,legend="MC"):
> display({p1,p2},title="Lattice Constant");
```



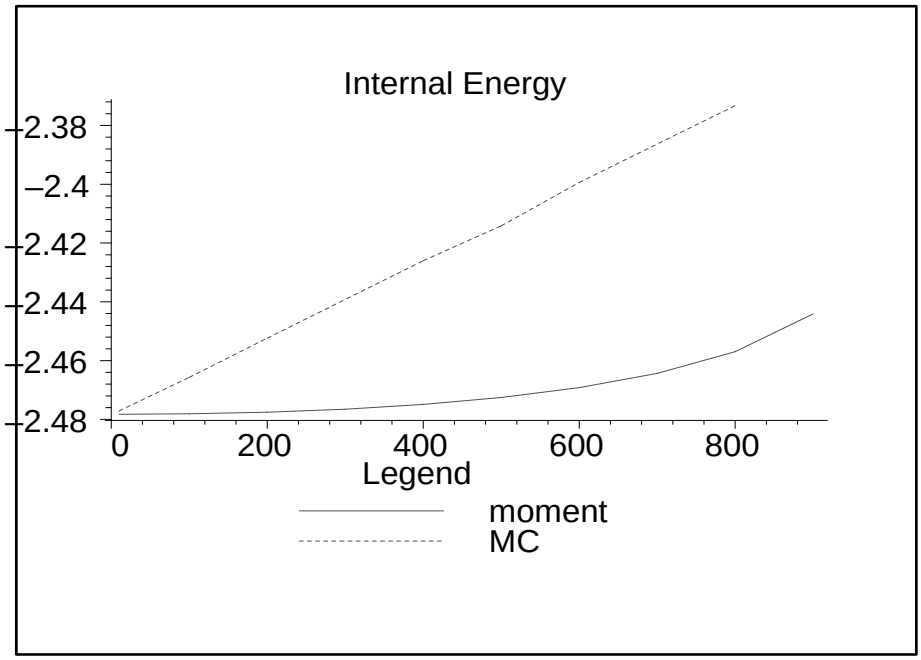

```
> M:=63.546:# M_Cu
> k:=5.43*2:#5.416914305;
> U0:=-2.4780:
> k_prime:=k*ergtoeV*10^(-7)/(1e-10)^2:#eV/A^2->J/m^2=(1/sec)
> M_prime:=M*10^(-3)/Avogadro:# Relative atomic mass -> real atom mass
> omega:=sqrt(k_prime/M_prime);
```

$$\omega := .4060731902\,10^{14}$$

```
> T:='T':#for repeated apply
> theta:= k_B*T:
> xx:= evalf(hbar*omega/(2*theta)):
> T_list:=[10,100,200,400,500,600,800]:
> B1:=[]:B2:=[]:B3:=[]:
> for i from 1 to nops(T_list) do
> T:=T_list[i];
> FF:=3*theta*(ln(1-exp(-2*xx))+xx)+U0;
> B1:=[op(B1),[T,B[i][2]]]:  #Internal Energy
> B2:=[op(B2),[T,FF]]: #Harmonic Free Energy
> B3:=[op(B3),[T,FF+B[i][5]]]:  #Total Free Energy
> od:
> p1:=plot(A3,linestyle=1,legend="moment"):
> p2:=plot(B2,linestyle=2,legend="MC_Harmonic Free"):
> p3:=plot(B3,linestyle=3,legend="MC_total Free"):
> display({p1,p2,p3},title="Total Free Energy");
```




```
> p1:=plot(A1,linestyle=1,legend="moment"):
> p2:=plot(B1,linestyle=2,legend="MC"):
> display({p1,p2},title="Internal Energy");
```



参考文献

[Salomons] E. Salomons, "hcp-bcc transition and the free energies of the hcp and bcc structures of zirconium" Phys. Rev. B, **43** (1991), 6167–6169.

[Kubo] 久保亮五編，大学演習熱学・統計力学，(1961 裳華房).

[Kamiyama and Sato] 神山新一，佐藤明，モンテカルロ・シミュレーション，(1997 朝倉書店).

[Ueda] 上田顕，コンピュータシミュレーション (-マクロな系の中の原子運動-)，(1990 朝倉書店).

[Allen and Tildesley] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, (1989, Oxford University Press, Oxford).

[Fortran] 杉江　日出澄　他，FORTRAN 77 による数値計算法，(1986 培風館).

[Bennett76] C. H. Bennett, "Efficient Estimation of Free Energy Differences from Monte Carlo data", J. Comp. Phys., **22** (1976), 245–268.

[Frenkel84] D. Frenkel and A. J. C. Ladd, "New Monte Carlo method to compute the free energy of arbitrary solids. Application to the fcc and hcp phases of hard spheres", J. Chem. Phys., **81**(7) (1984), 3188–3193;

[Frenkel86a] D. Frenkel, "Stability of the High-Pressure Body-Centered-Cubic Phase of Helium", Phys. Rev. Lett., **56**(8) (1986), 858–860;

[Frenkel86b] D. Frenkel, "Free-Energy Computation and First-Order Phase Transitions", *Proceedings of "Enrico Fermi", Molecular-dynamics simulation of statistical-mechanical systems:*, ed. by G. Ciccotti and W.G.Hoover, 1986, 151–188.