卒業論文

体積ーエネルギーカーブからの熱膨張の予測

関西学院大学　理工学部　情報科学科
2639　永山　優

2006年3月

指導教員　西谷滋人　教授

目 次

概 要

熱膨張は有限温度における格子振動の効果によって現れる．ところが，第一原理計算でこの振動の効果を取り入れて，有限温度で振動の効果を計算により求めていくには，ダイナミカルマトリックス, フォノン分散, フォノン DOS といったさまざまな段階を踏んで計算を行わなければならない.

そこで本研究では第一原理計算によるアプローチではなく，Moruzzi らが採用した Quasi-harmonic 近似法，すなわち体積エネルギーカーブから自由エネルギーを計算する手法を用いた.

最初に Moruzzi らの論文に掲載されている計算手順を再現し，このアプローチの正確さを確かめるため，彼らが計算した K の自由エネルギーを同様に計算で求めた．また，自由エネルギーの平衡原子間距離を用いることで熱膨張係数を計算した．それらを彼らの算出した値と比較してみると，同じ結果が再現され，計算の妥当性が確認された．そこで，この手法を Ti-bcc に適用し自由エネルギーを計算した結果，熱膨張を再現することができた.

第1章　緒言

熱膨張は有限温度における格子振動の効果によって現れてくる．第一原理計算でこの振動の影響を計算すると，ダイナミカルマトリックス, フォノン分散, フォノン DOS といったさまざまな段階を踏んで計算を行わなければ，この振動の影響を考慮できない.

そこで本研究では，第一原理計算による手法ではなく，Moruzzi らが採用した，体積ーエネルギーカーブから自由エネルギーを計算する手法，Quasi-harmonic 近似法を用いた．本研究の目的はこの手法を用いて，手軽に熱膨張を計算することである．本研究で実行したことは，まず Moruzzi らの論文に掲載されている計算手順を再現し，彼らが計算した K の自由エネルギーを Maple を用いて計算し，プロットした.

彼らがサンプルとして計算した K の熱膨張を，論文に掲載されている式を用いて計算し，結果を確かめた．その結果，彼らの計算結果と同じ値が算出できた．そこで，実際にこの手法を Ti-bcc に適用し自由エネルギーを計算した.
つまり，このプログラムを使えば，元素の物性値パラメータを入力するだけで，自由エネルギーが計算でき，今後違う物質の自由エネルギーを計算するにしても容易に計算できることになる．そして最後に，K，Ti-bcc の熱膨張係数を計算し，その様子をプロットした．グラフからも熱膨張している様子がはっきり確認できた.

第2章　熱膨張の起源

２原子が十分離れた距離でのエネルギーを基準として０とし，２原子を近づけていくと結合順位の低下に伴って，ある距離までエネルギーが減少する，また，近づきすぎると電子同士が重なり，反発する力が強くなり，エネルギーは急上昇する．固体でもこの２原子分子と同様に結合エネルギーの距離依存性が求められる．原子間ポテンシャルにより結合エネルギー曲線を求めると図2.1のようになる．熱膨張はこのような原子間ポテンシャルの２次以上の項によって現れる．

平衡点からの原子の変位 x のポテンシャルエネルギーを

$$U(x) = cx^2 - gx^3 \tag{2.1}$$

と取ることができる．

２次までの項では，調和振動子を表し，熱膨張は現れない．x^3 の項が原子間相互作用の非対称性を表し，この項が熱膨張係数と直接関わってくる．また有限温度での平均の変位は，ボルツマン分布関数を計算することで求まる．

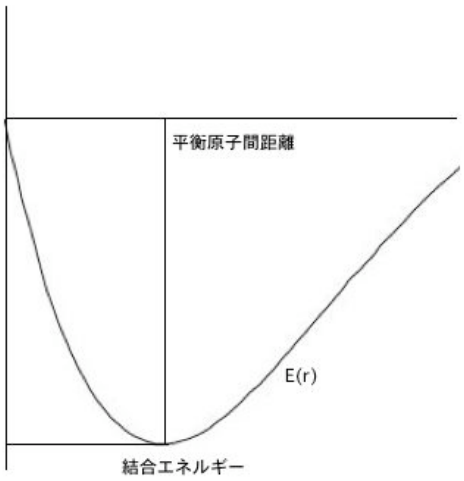

図 2.1: 結合エネルギーの原子間距離 x に対する依存性

第3章　Quasi-harmonic近似法

本章では，Moruzzi らが採用した Quasi-harmonic 近似法 [1] の手順を順次述べていく．

3.1　２体間ポテンシャル

3.1.1　Morse ポテンシャル

バネモデルは，個体のミクロの振る舞いを記述する最も適切なモデルである．個体のバネを数学的に記述する２体間ポテンシャル (pair potential) には，モース (Morse) 型がある．Morse ポテンシャルは，２原子分子の原子間距離 r に依存するポテンシャルである．

まず Moruzzi らが計算した K の結合エネルギーから物性値パラメータ $A,\quad D,\quad \lambda$，r_0 を以下の原子間ポテンシャルの式にフィットして求める．

結合エネルギーを

$$E(r) = a + be^{-\lambda\, r} + ce^{-2\,\lambda\, r} \tag{3.1}$$

と表す．

次に Morse ポテンシャルに結合エネルギーをフィットさせると以下のようになる．

$$E(r) = A - 2De^{-\lambda\,(r-r_0)} + De^{-2\,\lambda\,(r-r_0)} \tag{3.2}$$

これに以下の計算式を使うことで物性値パラメータが抽出される．

$$a = A, \tag{3.3}$$

$$\frac{b}{c} = -2e^{-\lambda\, r_0}, \tag{3.4}$$

$$D = \frac{b^2}{4c}. \tag{3.5}$$

これらのパラメータを使うことで，自由エネルギーを求める際に考慮しなければならない体積弾性率 B(bulk modulus) を計算することができる．

3.2 体積弾性率 B(bulk modulus)

体積弾性率とは物体に圧力を加えたときの変形のしにくさを示す指標である．これを体積と圧力の関係から導きだす．

まず，

$$x = e^{-\lambda\ r} \tag{3.6}$$

と置くと，式 (3.1) 式は

$$E = a + bx + cx^2 \tag{3.7}$$

と表せる．

熱膨張を計算するためには，圧力の体積および温度依存性を決定する必要がある．そこで圧力 P は全エネルギーから派生した負の体積を取ることで評価することができる。

$$P = -\frac{\partial\ E}{\partial\ V} \tag{3.8}$$

これを

$$P = -\frac{\partial\ E/\partial\ x}{\partial\ V/\partial\ x} \tag{3.9}$$

と変形させる．ここで体積 V は，

$$V = \frac{4}{3}\pi\ r^3 \tag{3.10}$$

と表せる．また，(3.6) より r は

$$r = -\frac{\ln x}{\lambda} \tag{3.11}$$

と表せることから，体積 V は

$$V = \frac{4\ \pi}{3}(-\frac{\ln x}{\lambda})^3 \tag{3.12}$$

というように x の関数と見ることができる．これを (3.9) 式に戻すと，圧力 P は以下のように定義できる．

$$P = \frac{x\ \lambda^3}{4\ \pi\ \ln x^2}(b + 2cx) \tag{3.13}$$

ここで，体積弾性率 B は

$$B = -V\frac{\partial\ P}{\partial\ V} \tag{3.14}$$

と表されるので，これも上記の P と同様に

$$B = -V\frac{\partial\ P/\partial\ x}{\partial\ V/\partial\ x} \tag{3.15}$$

(3.15) 式のように変形し，先ほど求めた (3.12)，(3.13) 式を (3.15) 式に代入すると，体積弾性率は

$$B = -\frac{x\ \lambda^3}{12\ \pi\ \ln x}((b+4cx) - \frac{2}{\ln x}(b+2cx)) \tag{3.16}$$

と表すことができる．この体積弾性率は次に記述するデバイ温度 $\Theta_{\rm D}$ に深く関係する物性値となる．

3.3 デバイ温度 $\Theta_{\rm D}$

デバイ温度とは格子振動や，結晶の結合力といったような物理量と深くかかわり合いを持つ温度のことを指し，物質の硬さの指標となる．一般的に物質のデバイ温度は，硬い物質ほど高く，逆に柔らかい物質ほど低い値を取る．

低温では低振動数の格子モードのみが基底状態から励起されることになる．[2] 基底状態とは，最もエネルギーの低い安定な状態のことを言う．低振動数のモードとしては，長波長の音響モード，つまり音波があり，その分散関係は

$$\omega = v_{\rm s}k \tag{3.17}$$

の形をとる．ここで v_s は音速を表す．
この分散関係にともなう状態密度関数 $g(\omega)$ は，

$$g(\omega) = \frac{Vk^2}{2\ \pi^2}\frac{dk}{d\ \omega} = \frac{V\ \omega^2}{2\ \pi^2 v_{\rm s}^3} \tag{3.18}$$

と表すことができる．ここでは $v_{\rm s}$ は伝搬の方向に依存し，ω は k の大きさだけによるという仮定は成立しない．しかし，もし因子 $1/v_{\rm s}^3$ が全ての方向にわたって平均したものを表していると見なすと，

$$g(\omega) = \frac{V\ \omega^2}{2\ \pi^2}(\frac{1}{v_l^3} + \frac{2}{v_t^3}) \tag{3.19}$$

と表すことができる．ここでの状態密度は 3 個の音響分岐，すなわち 1 個の縦波 (longitudinal) と 2 個の横波 (transverse) にわたって和をとる必要がある．ここで，v_t，v_l はそれぞれ縦波と横波のモードの音速であり，

$$\frac{1}{v_{\rm s}^3} = \frac{1}{v_l^3} + \frac{2}{v_t^3} \tag{3.20}$$

$v_{\rm s}$ は全ての方向にわたって平均したものとみなすことができる．

次に，状態密度関数をカットオフ振動数 $\omega_{\rm D}$ まで積分する．1 原子あたり，つまり 1 次元で考えているのを 3 次元で考えると，モードの数は 3 となり，

$$\int_0^{\omega_{\rm D}} g(\omega)d\ \omega = 3 \tag{3.21}$$

これを解くと，

$$\frac{V}{6\ \pi^2}(\frac{1}{v_l^3}+\frac{2}{v_t^3})\ \omega_{\rm D}^3=3 \tag{3.22}$$

と表される．ここで，3 個の音響分子の平均を

$$\frac{3}{v^3}=\frac{2}{v_t^3}+\frac{1}{v_l^3} \tag{3.23}$$

とし，(3.22) 式に代入すると，

$$\frac{V}{6\ \pi^2}v^{-3}\omega_D^3=1 \tag{3.24}$$

と変形することができる．また，v を ρ (密度) と B(体積弾性率) で定義すると，

$$v=\sqrt{\frac{B}{\rho}} \tag{3.25}$$

となる．また ρ は，

$$\rho\ =\frac{M}{V} \tag{3.26}$$

と表され，体積 V は，

$$V=\frac{4}{3}\pi\ r^3 \tag{3.27}$$

だから，(3.26) 式と (3.27) 式を (3.25) 式に代入すると，

$$v=\sqrt{\frac{4}{3}\pi\ r^3B} \tag{3.28}$$

というように変形できる．これを (3.24) 式に代入すると，

$$\omega_{\rm D}^3=6\ \pi^2\frac{3}{4\ \pi\ r^3}v^3=6\ \pi^2(\frac{4}{3}\pi\ r^3)^{-1}(\frac{4\ \pi\ r^3}{3})^{\frac{3}{2}}(\frac{B}{M})^{\frac{3}{2}} \tag{3.29}$$

となり，

$$\omega_{\rm D}=(6\ \pi^2)^{\frac{1}{3}}(\frac{4}{3}\pi\ r^3)^{\frac{1}{6}}\sqrt{\frac{B}{M}} \tag{3.30}$$

と 表すことができる．ここで，デバイ温度は，

$$k_{\rm B}\Theta_{\rm D}=\frac{h}{2\ \pi}\omega_{\rm D} \tag{3.31}$$

と一般的に表され，このときの $k_{\rm B}$，$\omega_{\rm D}$ はそれぞれ[1]ボルツマン定数 $k_{\rm B}$(Boltzmann's constants)，[2]プランク定数 h(Plank's constants) と呼ばれる定数のことを指す．ま

[1] $k_{\rm B}$=1.3806503 $\times$ 10^{-23}[J/K]
[2] h=6.6261 $\times$ 10^{-34}[J/s]

た，ω_{D} はデバイ振動数 (Debye frequency) である．これに式 (3.30) 式を代入すると，

$$\Theta_{\mathrm{D}} = \frac{h}{2\ \pi\ k_{\mathrm{B}}}(6\ \pi^{2})^{\frac{1}{3}}(\frac{4}{3}\pi\ r^{3})^{\frac{1}{6}}\sqrt{\frac{B}{M}} \tag{3.32}$$

これを計算すると，

$$\Theta_{\mathrm{D}} = 67.48\sqrt{\frac{rB}{M}} \tag{3.33}$$

となる．ここで，v_t，v_l をそれぞれ ρ を用いて表すと，

$$v_t = \sqrt{\frac{S}{\rho}} \tag{3.34}$$

$$v_l = \sqrt{\frac{L}{\rho}} \tag{3.35}$$

というように表され，これらに Molzzi らが計算した値，S=0.30B，L=1.42B をそれぞれ代入すると，v は

$$v = 0.617\sqrt{\frac{B}{\rho}} \tag{3.36}$$

となる．これを (3.30) 式に代入し，Θ_{D} を計算すると，

$$(\Theta_{\mathrm{D}})_0 = 41.63\sqrt{\frac{r_0 B}{M}} \tag{3.37}$$

となる．ここで，r_0 は平衡原子間距離 [a.u.]，M は原子量，B は r_0 において計算された体積弾性率 [kbar] を表す．以下，式 (3.37) を用いて自由エネルギーを計算する．

3.4 デバイ関数 $D(\Theta_{\mathrm{D}})$

デバイ関数とはデバイ温度をパラメータとする関数で

$$D(y) = \frac{3}{y^3}\int_0^y \frac{e^x x^4}{(e^x-1)^2}dx \tag{3.38}$$

と表せる．これは y $\to 0$ とすると，$D \to 1$ となり，古典統計力学に対応している．この関数の y=Θ_{D} として計算していく．

3.5 自由エネルギー F

自由エネルギーとは，振動している物質の全エネルギーである，これを式で表現すると，以下のようになる．

$$F(r,T) = E(r) + E_D(r,T) - TS_D(r,T) \tag{3.39}$$

ここで，T は温度，E_D と S_D はデバイ関数で，E_D と S_D はそれぞれ

$$E_D(r,T) = 3k_BTD(\Theta_D/T) + E_0, \tag{3.40}$$

と，

$$S_D(r,T) = 3k_B[\frac{4}{3}D(\Theta_{\rm D}/T) - \ln(1-e^{-\Theta_{\rm D}/T})] \tag{3.41}$$

となる．ここで E_0 は零点エネルギーを表し，

$$E_0 = \frac{9}{8}k_{\rm B}\Theta_{\rm D} \tag{3.42}$$

とされる．以上より自由エネルギーの最終形態として式をまとめると，以下のようにまとめることができる．

$$f(r,T) = E(r) - k_{\rm B}T[D(\Theta_{\rm D}/T) - 3\ln(1-e^{-\Theta_{\rm D}/T})] + \frac{9}{8}k_{\rm B}\Theta_{\rm D} \tag{3.43}$$

これまで考慮してきた，体積弾性率，デバイ温度，デバイ関数をこの自由エネルギーの式に当てはめることで，容易に物質の自由エネルギーを計算することができ，さらに熱膨張も計算することができるようになる．

3.6 熱膨張係数 α

この求めた自由エネルギーから，各温度における自由エネルギーの最小点をとり，それをつなげていくと，r_0[a.u.] と T[K] の関数で表すことができる．これを示したものが，

$$\alpha = \frac{1}{r_0}\frac{dr_0}{dT} \tag{3.44}$$

となる．つまり，自由エネルギーの各温度における平衡原子間距離さえわかれば，熱膨張係数が計算できることになる．

第4章　結果

まず，最初にこの論文に記載されている計算式の正確性を立証するために，実際にKの自由エネルギーを計算してみた．計算を実行していくにつれ，いくつかの記述の誤りがあり，それを訂正することで，論文に掲載されているものと同様の結果が得られるようになった．

苦労した点は，自分の実行したプログラムと論文の結果とでは，体積弾性率と自由エネルギーの単位が合わず，自由エネルギーの式を完全に再現しても同じグラフがでなかった．そこで，体積弾性率の単位を[1] [Ry/a.u.^{3}], [eV/Å^3], [GPa], [kbar]と変化させることでうまく対応できた．

また自由エネルギーに関してはRyとボルツマン定数k_{B}を使い，単位をmRyに合わせることで正確な結果が抽出できた．以下に，実際計算したプログラムとそのグラフをKとTi-bccに分けて掲載していく．

4.1　K

Moruzziらがサンプルとして記述したKにおける自由エネルギーを再現するために，彼らの実験結果から抽出された物性値パラメータを用いて，体積弾性率，デバイ温度，デバイ関数を計算していき，最後にこれを自由エネルギーの式に代入して，その様子をプロットした．

第３章の冒頭で記したように，最初単位が合わず，グラフが正確に抽出できなかったが，ボルツマン定数や単位をきちんと考慮することで解決できた．またデバイ関数に関してはこの論文には正確な式が掲載されていないので注意する必要がある．

プロットした自由エネルギーのグラフから平衡原子間距離を読み取ると，温度を上げるにつれて変化していく様子がはっきりわかる．つまり，熱膨張が起こっていることが読み取れる．

実際プロットしたグラフの温度は　T=100, 200, 400, 800[K] と変化させた．以下に述べるTi-bccについても同じ温度で検証する．

また，この際求めた，各温度における自由エネルギーと，温度の関係式を使うことで熱膨張係数を計算した．

[1] 1a.u.=2Ry=27.2eV, 1a.u.=0.529177 Å, 1kbar=10^8Pa, Pa=2.94×10^3a.u., 1GPa=10kbar

4.1.1 ２体間ポテンシャル

まず，K の物性値パラメータは計算結果が残っているので，
A=-1196.377, D=0.0724, λ=0.6303, r_0=4.7719 を用いて K の結合エネルギーを求める．

これを実行した Maple のプログラムを掲載する．

```
> restart;
> lambda:=0.6303;
> A:=-1196.377;
> r0:=4.7719;
> De:=0.0724;
   lambda := 0.6303
   A := -1196.377
   r0 := 4.7719
   De := 0.0724
```

//A，r0，De(D)，lambda(λ) は V. L. Moruzzi らの論文に掲載されている値を使用．

```
> solve({De=b^2/(4*c),b/c=-2*exp(-lambda*r0),a=A},{a,b,c});
     {a = -1196.377000, b = -2.930950494, c = 29.66322789}
```

//a，b，c の単位は [Ry/atom]，
//M: 原子量

```
> a:=A;
> b:=-2.930950494;
> c:=29.66322789;
> M:=39.102;
> a := -1196.377
   b := -2.930950494
   c := 29.66322789
   M := 39.102
```

//このパラメータを使って結合エネルギーを計算すると

```
> E1:=(a+b*exp(-lambda*r)+c*exp(-2*lambda*r));
> E:=unapply(E1,r);
E1 := -1196.377 - 2.930950494 exp(-0.6303 r)
```

```
 + 29.66322789 exp(-1.2606 r)
E := r -> -1196.377 - 2.930950494 exp(-0.6303 r)
+ 29.66322789 exp(-1.2606 r)
```

となる．この結合エネルギーを図示すると以下の図 4.1 のようになる．

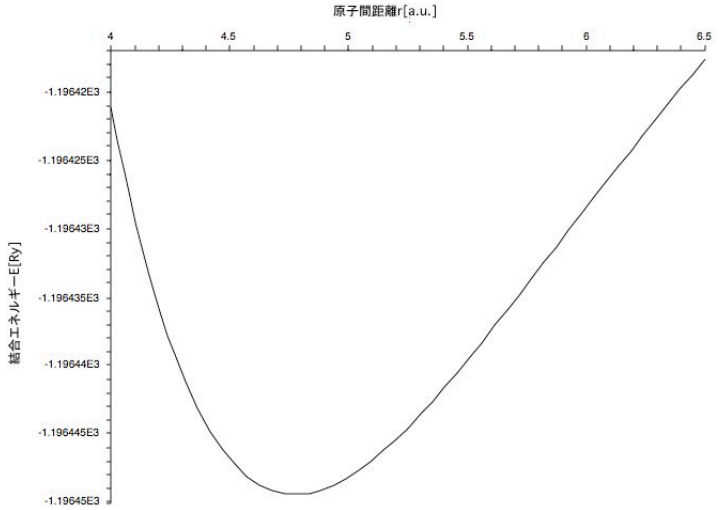

図 4.1: K の結合エネルギー E[Ry]

4.1.2 体積弾性率 B

体積弾性率は (3.16) 式で計算できる．これを計算したプログラムが以下の通りである．

```
//体積弾性率 (bulk modulus)    B=B1*B2:

> B1:=unapply
((-lambda^3*exp(-lambda*r))/(12*Pi*ln(exp(-lambda*r))),r);
> B2:=unapply
((b+4*c*exp(-lambda*r))-(2/ln(exp(-lambda*r)))*(b+2*c*exp(-lambda*r)),r);
> B:=unapply(B1(r)*B2(r)*((27.2/2)/(0.529177)^3)*160.218*10,r):

//r0 における B[kbar]
```

```
> evalf(B(r0));
                                47.02058827
```

これを図示したものが図 4.2 である.

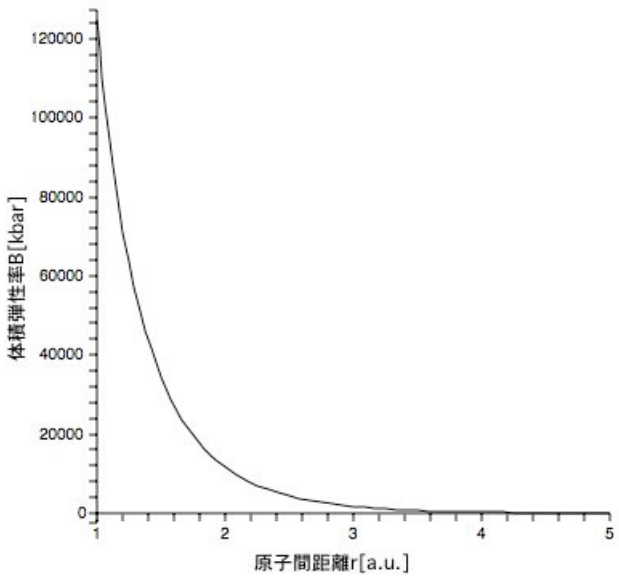

図 4.2: K における体積弾性率 B[kbar]

4.1.3 デバイ温度 Θ_{D}

続いてデバイ温度だが，式 (3.37) で表されるため．これに体積弾性率と K の原子量を代入して，r の関数としてプロットしてみると，図 4.3 のようになる．
その Maple でのプログラムは，

```
//デバイ温度 [K] (数百 K 程度)

> thetaD:=unapply(41.63*((r0*B(r))/M)^(1/2),r):
```

となる.

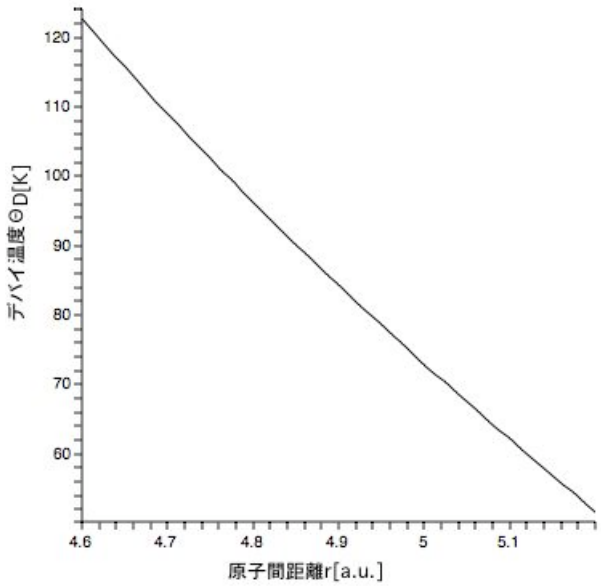

図 4.3: K におけるデバイ温度 Θ_{D}[K]

4.1.4 デバイ関数 $D(\Theta_{\mathrm{D}})$

デバイ関数はデバイ温度をパラメータとした関数である．この関数は (3.38) 式のようになる．実際に Maple で書いてみると，

```
> Debye:=unapply((3/y^3)*Int(exp(x)*x^4/(exp(x)-1)^2,x=0..y),y):
> Df:=unapply(Re(evalf(Debye(thetaD(r)/T))),r,T):
```

となる．
このプログラムを実行すると，実数値と虚数値が現れてしまうので，Re で実数だけに限定している．これをプロットしたものが図 4.4 となる．

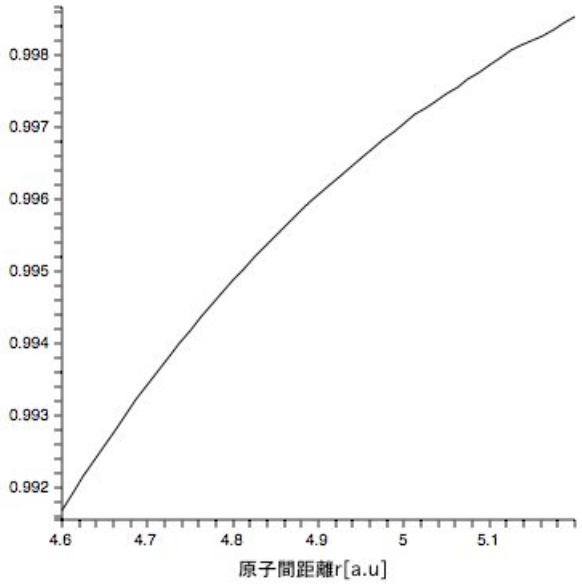

図 4.4: T=300 としたときの K のデバイ関数 D(Θ_{D})

4.1.5 自由エネルギー F

自由エネルギーは以下のようにして求まる.

```
> eq1:=unapply((kb/Ry)*T*(Df(r,T)-3*ln(1-exp(-thetaD(r)/T)))
-(9/8)*(kb/Ry)*thetaD(r),r,T):

//(kb/Ry) を掛けることで単位を Ry に変換している.
//まず E(r0) のエネルギーを基準値 (= 0) とし,
//次に自由エネルギーの単位を mRy にするため全体に m(ミリ: 10^3) を掛けた.

> g:=(-E(r0)+E(r)-eq1(r,T))*10^3:
> f:=unapply(g,r,T):

//f: 自由エネルギーの式

> with(plots):
Warning, the name changecoords has been redefined
> p1:=plot(f(r,100),r=4.6..5.2,color=black):
> p2:=plot(f(r,200),r=4.6..5.2,color=black):
> p3:=plot(f(r,400),r=4.6..5.2,color=black):
```

```
> p4:=plot(f(r,800),r=4.6..5.2,color=black):
```

//r=4.6~5.2の範囲で温度Tを100，200，400，800と変化させたものをグラフ化した．

```
> display(p1,p2,p3,p4);
evalf(f(r0,100),6);
evalf(f(r0,200),6);
evalf(f(r0,400),6);
evalf(f(r0,800),6);
```

```
                    -0.767930
                     -4.08986
                     -13.6579
                     -43.6881
```

これを実際にプロットしたものが図 4.5 となる．

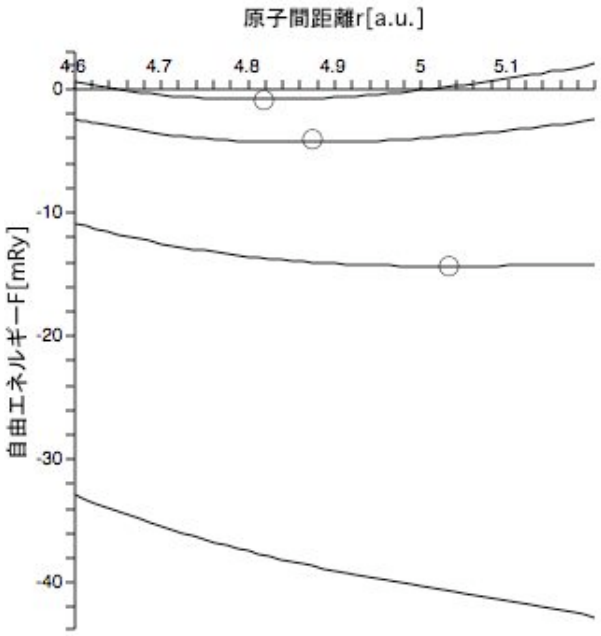

図 4.5: T=100, 200, 400, 800[K] としたときの K の自由エネルギー [mRy]

図 4.5 を見ると，温度を上げるにつれて平衡原子間距離が変化していく様子がはっきりわかる．つまり，熱膨張がはっきりと確証できる．ちなみに，T=100, 200, 400, 800[K] における平衡原子間距離の K の自由エネルギーの値と平衡原子間距離の値を記述したものが表 4.1 となる．しかし，図 4.6 のように，800[K] における自

表 4.1: 温度を変化させたときの K の平衡原子間距離とそのときの自由エネルギーの値

T[K]	平衡原子間距離 r_0[a.u.]	自由エネルギー F[mRy]
100	4.8171	-0.81835
200	4.8656	-4.33144
400	5.0291	-14.3131
800	———	———

由エネルギーの平衡点が現れない．これは K の融点が低く，800[K] まで温度を上げてしまうと，融けてしまうからである．

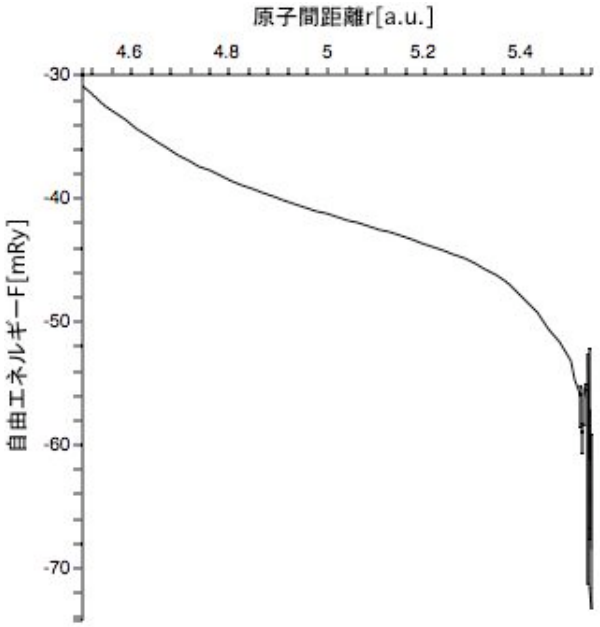

図 4.6: T=800 における K の自由エネルギー

これらより，K の自由エネルギーは正確に再現することができ，彼らの手法の正確さが確認された．よって，この手順にしたがって Ti-bcc についても同様に自由エネルギーを計算していく．

4.1.6 熱膨張係数α

熱膨張係数は (3.44) 式で計算することができる．これを Maple で書いたものが以下である．

```
> with(stats):
> with(linalg):
> with(plots):
> p1:=4.6:p2:=5.2:

//範囲指定 4.6〜5.2

> n_div:=(p2-p1)/20:
> for T1 from 1 to 30 do
> data[T1]:=[];datax[T1]:=[];datay[T1]:=[];
> for i1 from 0 to 20 do
> rr:=p1+n_div*i1;
> #printf("%10.5f %10.5fn",rr,f(rr,10*T1));
> datax[T1]:=[op(datax[T1]),rr];
> datay[T1]:=[op(datay[T1]),evalf(f(rr,10*T1))];

//datax[T1]: x 座標，datay[T1]: y 座標
//10*T1 で温度を 10K ずつとる．

> end do:
> data[T1]:=[datax[T1],datay[T1]];
> #print(data[T1]);
> end do:

Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform

> for T1 from 1 to 30 do
> P:=transpose(data[T1]):
> pointplot(P):
> d[T1]:=pointplot(P):
> #print(d[T1]);
> end do:
> for i from 1 to 30 do
> with(stats):
```

```
> fit[i]:=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}]](data[i]):
```

//x と y のデータ [T1] をフィッティングする.

```
> f||i:=unapply(rhs(fit[i]),x);
> end do:
```

```
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
: :
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
```

```
> data2:=[]:
> for i from 1 to 30 do
> data2:=[op(data2),[10*i,fsolve(diff(f||i(x)=0,x),x=4..5)]];
```

//微分して＝0 となる x の値を計算し，配列に格納する．この値を x’ とす
ると
//配列の中身は [T(10K ずつ) , x’] となる.

```
> end do:
> data2:
> pointplot(data2);
```

```
> data3:=convert(transpose(convert(data2,array)),listlist):
> with(stats):
> fit100:=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):
```

```
//data2をフィッティング.

> fa:=unapply(rhs(fit100),x):
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
> d100:=plot(fa(x),x=0..300):
> dr:=pointplot(data2):
> display(d100,dr);

> faa:=diff(fa(x),x);

faa := 0.00004716321561 + 0.000006679377244 x - 4.464971202 10^(-8) x^2
 + 1.468506974 10^(-10)  x^3  - 1.661259842 10^(-13)  x^4
> daa:=plot(faa(x),x=0..300):
> display(daa);

> faaa:=r0^(-1)*faa;
> daaa:=plot(faaa(x),x=0..300):

//熱膨張係数

> display(daaa);

faaa := 0.000009883529750 + 0.000001399731186 x - 9.356799603 10^(-9) x^2
   + 3.077405172 10^(-11) x^3 - 3.481338339 10^(-14) x^4
```

これらから，K の熱膨張係数をプロットすると，図 4.7 となる.

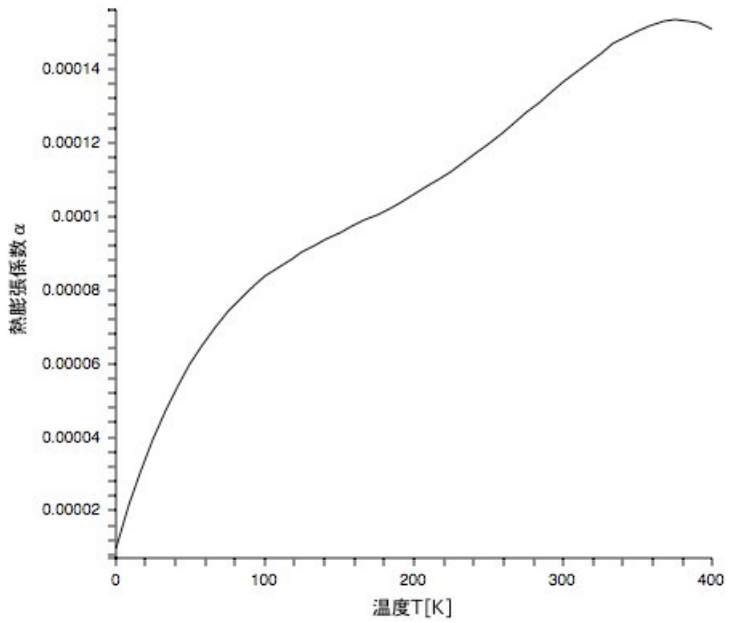

図 4.7: T=0-400[K] としたときの K の熱膨張係数

4.2 Ti-bcc

Ti-bcc の結合エネルギーを計算する場合，K と違って物性値パラメータは与えられていないので，同研究室員の真田氏が第一原理計算で求めた Ti の結合エネルギーを使って，物性値パラメータを計算した．

注意した点は，真田氏が計算した Ti の結合エネルギーの単位が [eV] だったため，[Ry] に変換しなければいけないことであった．自由エネルギーを計算するまでの過程は基本的に K の場合と同様である．

また，Ti の自由エネルギーをプロットしてみると (K と同様に温度は T=100, 200, 400, 800[K] と変化させた)，温度を上げていくにつれて，Ti-bcc の自由エネルギーの平衡原子間距離が少しずつ変化していく様子が伺える．しかし，K と比較すると変化量は大きくない．これは K と Ti-bcc とでは体積弾性率に大きな差があり，体積弾性率の大きさは変形のしにくさを示すことなどから予想される結果である．

4.2.1　２体間ポテンシャル

TI-bcc も同様の手法で自由エネルギーを計算していく．
まず，真田裕示氏が第一原理計算で計算した Ti-bcc の結合エネルギーの値を使う．これ原子間ポテンシャルにフィットさせて， 物性値パラメータ A, λ , D, r_0 を抽出する．これを実際に Maple で実行していくと，以下のようになる．

```
> restart;
p1 := [[2.61, -9.307376], [2.639, -9.94744], [2.726, -11.600086],
 [2.9, -13.878388], [3.045, -14.893112], [3.103, -15.115222],
 [3.19, -15.284125], [3.2, -15.292761], [3.3, -15.260682],
 [3.5, -14.715456], [4.5, -9.892994]];
for i from 1 to nops(p1) do
  p1[i][2]:=p1[i][2]/2;
end do:

//p1=[ 格子定数 [Å], 結合エネルギー [eV] ]

> with(linalg):
> data11:=convert(transpose(convert(p1,array)),listlist);
> data11[1];
> data1:=[data11[1]*evalf((3)^(1/2)/2),data11[2]];
> q1:=convert(transpose(convert(data1,array)),listlist);
data11 := [[2.61, 2.639, 2.726, 2.9, 3.045, 3.103, 3.19, 3.2, 3.3, 3.5, 4.5],
  [-4.653688000, -4.973720000, -5.800043000, -6.939194000,
 -7.446556000, -7.557611000, -7.642062500, -7.646380500,
-7.630341000, -7.357728000, -4.946497000]]
[2.61, 2.639, 2.726, 2.9, 3.045, 3.103, 3.19, 3.2, 3.3, 3.5, 4.5]

data1 := [[2.260326304, 2.285441041, 2.360785251, 2.511473672,
2.637047355, 2.687276829, 2.762621039, 2.771281293, 2.857883833,
3.031088914, 3.897114318 ],
[-4.653688000, -4.973720000, -5.800043000, -6.939194000, -7.446556000,
-7.557611000, -7.642062500, -7.646380500, -7.630341000, -7.357728000,
 -4.946497000]]

q1 := [[2.260326304, -4.653688000], [2.285441041, -4.973720000],

  [2.360785251, -5.800043000], [2.511473672, -6.939194000],
```

```
  [2.637047355, -7.446556000], [2.687276829, -7.557611000],

  [2.762621039, -7.642062500], [2.771281293, -7.646380500],

  [2.857883833, -7.630341000], [3.031088914, -7.357728000],

  [3.897114318, -4.946497000]]

> with(stats):
> fit1:=fit[leastsquare[[x,y], y=a+b*x+c*x^2+d*x^3+e*x^4+f*x^5]](data1):
> f2:=unapply(rhs(fit1),x);
Warning, these names have been redefined: anova, describe,
 fit, importdata, random,
statevalf, statplots, transform

f2 := x -> 183.0109077 - 175.3991853 x + 44.16056438 x^2  +
 2.588249377 x^3  - 2.296271103 x^4  + 0.2240169948 x^5
> with(plots):
> pp1:=pointplot(q1):
> f1:=(a,b,c,d,r)->a+b*exp(-d*r)+c*exp(-2*d*r);
    f1 := (a, b, c, d, r) -> a + b exp(-d r) + c exp(-2 d r)
> pp2:=plot(f2(r),r=2.6..4.5):
> display(pp1,pp2);

> x0:=fsolve(diff(f2(x),x),x=2.8..4.4);
> y0:=f2(x0);
> y1:=subs(x=x0,diff(f2(x),x));
> y2:=subs(x=x0,diff(f2(x),x,x));
> y3:=subs(x=x0,diff(f2(x),x,x,x));
                        x0 := 2.800614869
                        y0 := -7.65827074
                        y1 := 1.5 10^(-7)
                        y2 := 14.10242002
                        y3 := -33.3899271
> eqs:={
  y0=f1(a,b,c,d,x0),
  subs(x=x0,diff(f1(a,b,c,d,x),x))=0,
  y2=subs(x=x0,diff(f1(a,b,c,d,x),x,x)),
```

```
  y3=subs(x=x0,diff(f1(a,b,c,d,x),x,x,x))
  }:
> sol1:=solve(eqs,{a,b,c,d});
sol1 := {b = -206.4522629, c = 941.2755883, a = 3.662148431,
 d = 0.7892245220}
> f3:=unapply(subs(sol1,f1(a,b,c,d,x)),x);
   f3 := x -> 3.662148431 - 206.4522629 exp(-0.7892245220 x)

      + 941.2755883 exp(-1.578449044 x)
> pp3:=plot(f3(r),r=2.6..4.5,color=blue):
> display(pp1,pp2,pp3);

> E1:=(2/27.2)*f3(x);
        E1=0.2692756199 - 15.18031345 exp(-0.7892245220 x)

            + 69.21144031 exp(-1.578449044 x)
```

ここで注意すべき点は，真田氏が計算した Ti-bcc の結合エネルギーは [eV] で計算されているため，単位を [Ry] に変換する必要があった．また，ここで計算した r_0 は E1 における平衡点，すなわちエネルギーが一番大きいときの原子間距離を表している．実際にこの結合エネルギーをプロットしたものが図 4.8 となる．

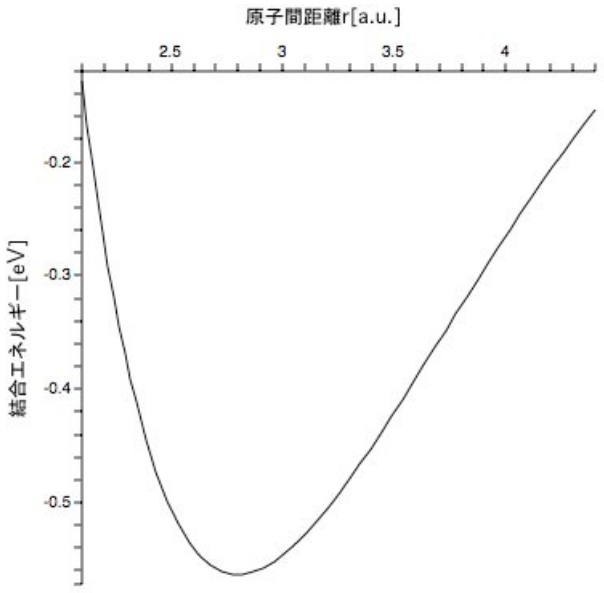

図 4.8: Ti-bcc の結合エネルギー E[Ry]

4.2.2 体積弾性率 B

次に体積弾性率だが,

```
//体積弾性率 (bulk modulus)    B=B1*B2:
> B1:=unapply((-lambda^3*exp(-lambda*r))/(12*Pi*ln(exp(-lambda*r))),r):
> B2:=unapply((b+4*c*exp(-lambda*r))-(2/ln(exp(-lambda*r)))
*(b+2*c*exp(-lambda*r)),r):
> B3(r):=B1(r)*B2(r)*91.77745886*160.218*10:

> B:=unapply(B3(r),r):
> plot(B(r),r=1..5,color=black);
> evalf(B(r0));

                              1443.924816
```

これをプロットしてみると，図 4.9 となる.

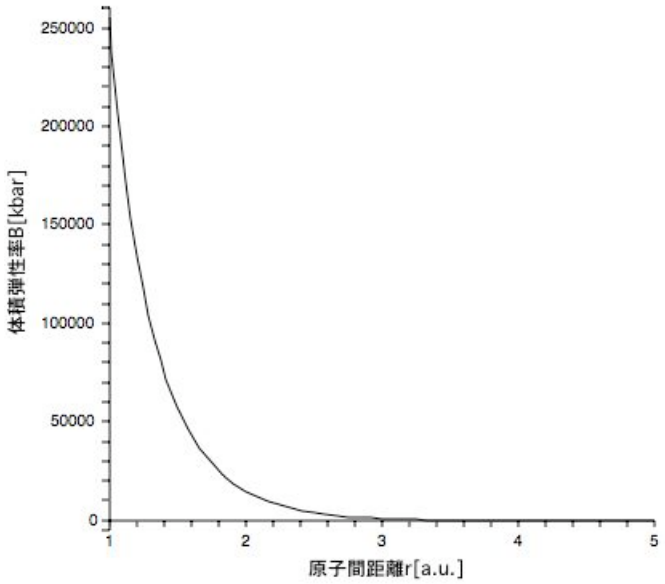

図 4.9: Ti-bcc の体積弾性率 B[kbar]

4.2.3 デバイ温度 Θ_{D}

次はデバイ温度であるが，これも同様に計算すると，以下のようになり，図 4.10 のようになる.

```
//デバイ温度 [K] （数百 K 程度）
> thetaD:=unapply(41.63*(r*B(r)/M)^(1/2),r):

> plot(thetaD(r),r=2.65..2.75,labels=["r[a.u.]","T[K]"],
labeldirections=[HORIZONTAL,VERTICAL]);

//デバイ温度のグラフと r0 におけるその値
> evalf(thetaD(r0));
382.5853407
```

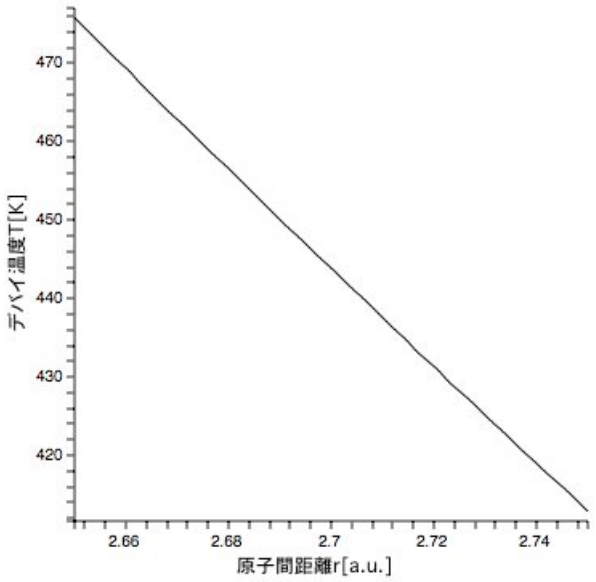

図 4.10: Ti-bcc のデバイ温度 Θ_{D}

4.2.4 デバイ関数 $D(\Theta_{\mathrm{D}})$

デバイ関数に関しても以下同様である．図 4.11 に示す．

```
//デバイ関数
> Debye:=unapply((3/y^3)*int(f1(x),x=0..y),y):
> Df:=unapply(Re(evalf(Debye(thetaD(r)/T))),r,T):
> plot(Df(r,300),r=2.75..2.85,color=black);
```

```
//T=300 の時のデバイ関数の値.
> Df(r0,300);
                                0.9231791521
```

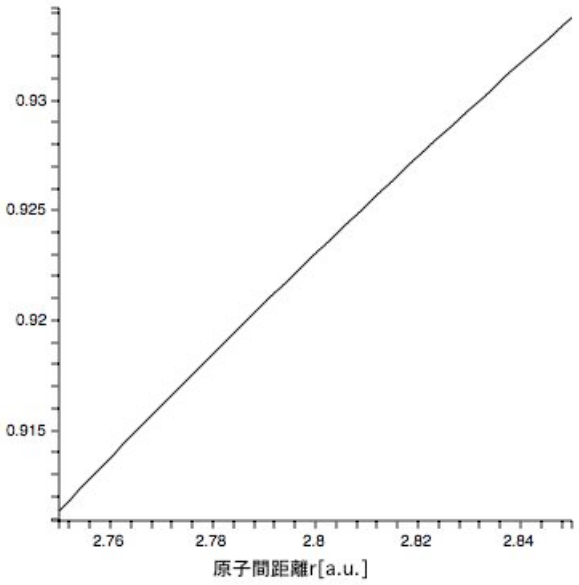

図 4.11: Ti-bcc の T=300 のときのデバイ関数 $\mathrm{D}(\Theta_{\mathrm{D}})$

4.2.5 自由エネルギー F

次に，自由エネルギーの式に体積弾性率，デバイ温度，デバイ関数を代入すると，図 4.12 のように自由エネルギーが抽出される.

```
> eq1:=unapply((kb/Ry)*T*(Df(r,T)-3*ln(1-exp(-thetaD(r)/T)))
-(9/8)*(kb/Ry)*thetaD(r),r,T):

//まず E(r0) のエネルギーを基準 (= 0) とし，
//次に自由エネルギーの単位を mRy にするため全体に m(ミリ: 10^3) を掛けた.

> g:=(-E(r0)+E(r)-eq1(r,T))*10^3:
> f:=unapply(g,r,T):

> with(plots):
```

```
Warning, the name changecoords has been redefined
> p2:=plot(f(r,100),r=2.75..2.85,color=red):
> p3:=plot(f(r,200),r=2.75..2.85,color=green):
> p5:=plot(f(r,400),r=2.75..2.85,color=blue):
> p6:=plot(f(r,800),r=2.75..2.85,color=black):
>
```

r=3.1-3.4 の範囲で温度 T を 100，200，400，800 と変化させたものをグラフ化した．

```
> display(p1,p2,p3,p4,labels=["r[a.u.]","Free energy[mRy]"],
labeldirections=[HORIZONTAL,VERTICAL]);
> evalf(f(r0,100),6);
> evalf(f(r0,200),6);
> evalf(f(r0,400),6);
> evalf(f(r0,800),6);
```

//r=r0 における自由エネルギー

```
0.730008
-1.35090
-7.81737
-26.1034
```

また，各温度における自由エネルギーの原子間距離と，そのときの値を示したものは表 4.2 である．

表 4.2: 温度を変化させたときの Ti-bcc の平衡原子間距離とそのときの自由エネルギーの値

T[K]	平衡原子間距離 r_0[a.u.]	自由エネルギー F[mRy]
100	2.8063	2.33670
200	2.8079	1.03553
400	2.8125	-3.44397
800	2.8233	-17.23733

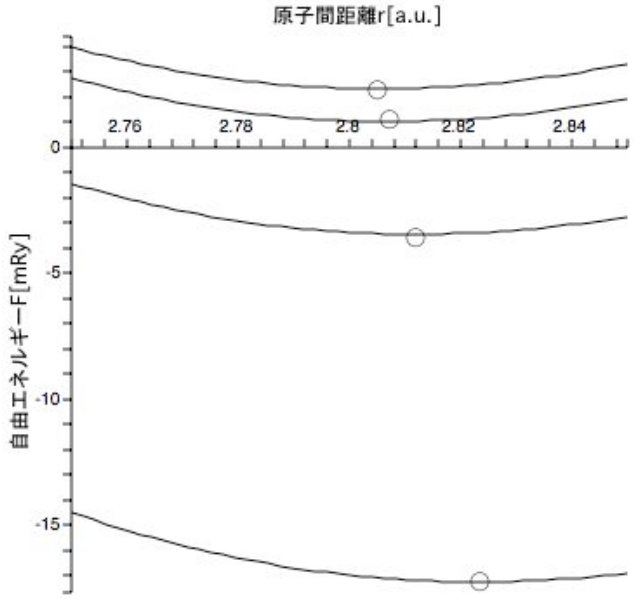

図 4.12: T=100,200,400,800[K] における Ti-bcc の自由エネルギー

4.2.6 熱膨張係数α

横軸に温度T，縦軸に平衡原子間距離r_0を取る．これをTで微分し，r_0で割ったものが熱膨張係数となる．それを実行したものが以下のプログラムであるが，Kのケースとほぼ同じであるので，説明は省略する．

```
> with(stats):
> with(linalg):
> with(plots):
> p1:=2.75:p2:=2.85:

//p1，p2で原子間距離の範囲指定
> with(stats):
> with(linalg):
> with(plots):
> p1:=2.75:p2:=2.85:
> n_div:=(2.85-2.75)/20:
> for T1 from 1 to 80 do
> data[T1]:=[];datax[T1]:=[];datay[T1]:=[];
> for i1 from 0 to 20 do
```

```
> rr:=p1+n_div*i1;
> #printf("%10.5f %10.5fn",rr,f(rr,10*T1));
> datax[T1]:=[op(datax[T1]),rr];
> datay[T1]:=[op(datay[T1]),evalf(f(rr,10*T1))];
> end do:
> data[T1]:=[datax[T1],datay[T1]];
> #print(data[T1]);
> end do:
>
Warning, the protected names norm and trace have been redefined and unprotecte
> for T1 from 1 to 80 do
> P:=transpose(data[T1]):
> pointplot(P):
> d[T1]:=pointplot(P):
> #print(d[T1]);
> end do:
> for i from 1 to 80 do
> with(stats):
> fit[i]:=fit[leastsquare[[x,y], y=c0+c1*x+c2*x^2+c3*x^3+c4*x^4]](data[i]):
> f||i:=unapply(rhs(fit[i]),x);
> end do:
>
>
Warning, these names have been redefined: anova, describe, fit, importdata,
 random, statevalf, statplots, transform
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
:
:
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
Warning, these names have been redefined: anova, describe, fit, importdata,
random, statevalf, statplots, transform
> d1:=plot(f1(x),x=2.75..2.85,color=black):
> d2:=plot(f2(x),x=2.75..2.85,color=black):
```

```
> d3:=plot(f3(x),x=2.75..2.85,color=black):
> display(d[1],d1);
> display(d[2],d2);
> display(d[3],d3);



> data2:=[]:
> for i from 1 to 80 do
> data2:=[op(data2),[10*i,fsolve(diff(f||i(x)=0,x),x=2.75..2.85)]];
> end do:
> data2:
> pointplot(data2);
> data2[10];
> data2[20];
> data2[30];
> data2[40];
> data2[50];
> data2[60];
> data2[70];
> data2[80];
                        [100, 2.806265423]
                        [200, 2.807871343]
                        [300, 2.810055751]
                        [400, 2.812519275]
                        [500, 2.815118150]
                        [600, 2.817798002]
                        [700, 2.820520927]
                        [800, 2.823277326]
> data3:=convert(transpose(convert(data2,array)),listlist):
> with(stats):
> fit100:=fit[leastsquare[[x,y],
y=c0+c1*x+c2*x^2+c3*x^3+c4*x^4+c5*x^5]](data3):
> fa:=unapply(rhs(fit100),x);
Warning, these names have been redefined: anova, describe, fit, importdata,
 random, statevalf, statplots, transform

fa := x -> 2.804720033 + 0.000002818165522 x + 5.850975438 10^(-8)  x^2
   - 7.211587401 10^(-11)   x^3  + 4.283966469 10^(-14)   x^4
```

```
- 8.710965713 10^(-18)   x^5

> limit(diff(fa(x),x),x=0);
> limit(diff(fa(x),x,x),x=0);
                        0.000002818165522
                                         -7
                         1.170195088 10
> d100:=plot(fa(x),x=0..800,color=black):
> dr:=pointplot(data2):
> display(d100,dr);

> faa:=diff(fa(x),x);

faa := 0.000002818165522 + 1.170195088 10^(-7)   x
- 2.163476220 10^(-10)   x^2 + 1.713586588 10^(-13)   x^3
- 4.355482856 10^(-17)   x^4

> daa:=plot(faa(x),x=0..800,color=black):
> display(daa);

> faaa:=r0^(-1)*faa;
> daaa:=plot(faaa(x),x=0..800,color=black):
> display(daaa);

faaa := 0.000001006266714 + 4.178350622 10^(-18)   x
- 7.725004405 10^(-11)   x^2 + 6.118608478 10^(-14)   x^3
- 1.555188078 10^(-17)   x^4
```

これを実行したものが図 4.13 となる.

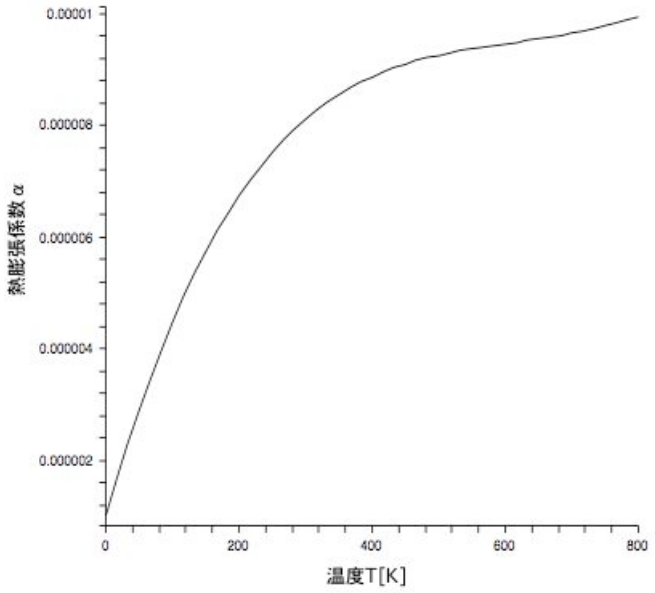

図 4.13: T=0-800[K] における Ti-bcc の熱膨張係数 α

第5章　総括

本研究では V. L. Moruzzi らが採用した，体積ーエネルギーカーブから自由エネルギーを計算する手法である Quasi-harmonic 近似法を用いて自由エネルギー及び，熱膨張係数を計算した.

しかし第一章で先述したように，彼らが掲載してる式にいくらかの誤りが見うけられた．一方，論文に記載されている計算手順は合っていたので，それらの誤りを訂正し，正しい計算式を用いて K の自由エネルギーを計算すると，彼らが検出した値と同じ値が検出された．つまり，彼らの計算結果を正確に再現できたことになる.

また次に，Ti-bcc の自由エネルギーにおいては，西谷研の真田氏が第一原理計算で計算した Ti-bcc の結合エネルギーを用いて計算を実行していった．体積弾性率，デバイ温度，デバイ関数，自由エネルギーと計算した結果，K 同様に Ti-bcc でも温度をあげるにつれて，自由エネルギーの平衡点の変位が見られ，また実際に熱膨張係数を計算し，様子をプロットするとはっきり熱膨している様子を確認することができた.

よって，本研究で作成した熱膨張を計算するプログラムを用いれば，K，Ti-bcc 以外の元素に関しても，結合エネルギーのデータさえ存在すれば，容易に自由エネルギー，熱膨張が計算できることになる.

謝辞

本研究を遂行していくにあたり，終始多大なる有益なご指導，及び丁寧な助言を頂いた西谷滋人教授に深い感謝の意を表します．

また，本研究を進めるにつれ，西谷研究室員の皆様にもさまざまな知識の供給，御協力を頂き，本研究を大成することができました．最後になりましたが，この場を借りて心から深く御礼申し上げます．

引用文献

[1] V. L. Moruzzi, J. F. Janak and K. Schwarz, "*Calculated thermal properties of metals*," Phys. Rev. B, **37**(1988), 790-799

[2] J. R. Hook and H. E. Hall 著, 「固体物理学入門 上」, 福山秀敏 監訳, 松浦民房, 鈴村順三, 黒田義浩 共訳, (丸善 (株), 東京, 2002).