卒業論文

小傾角粒界のシミュレーション

関西学院大学理工学部
情報科学科 西谷研究室 7644 八幡　裕也

2012年3月

指導教員　西谷　滋人　教授

概 要

本研究では粒子シミュレーションを作成し大槻の Al の粒界エネルギーに関する実験結果との比較を行った．また，作成したシミュレーションプログラムを Ruby から C 言語に書き換え計算の高速化を試みた．

粒界エネルギーは従来，Read-Shockley の式によって求まるとされている．しかし，近年，大槻は自らの実験データの角度依存性の傾きと一致しないという矛盾を示唆した．本研究は大槻が示唆した矛盾を粒子シミュレーションから解消することを目的としている．

粒界エネルギーの計算に用いるポテンシャルとして Lennard-Jones ポテンシャル，Morse ポテンシャル,EAM ポテンシャルの 3 種類のポテンシャルで検証を行った．その結果，EAM ポテンシャルを採用した．EAM ポテンシャルの様々なパラメータの値に対して，得られる結果を調べた．その結果，本シミュレーションに適したパラメータの値を見つけ出した．更に得られたパラメータを用いて様々な角度での計算を行い，実際にシミュレーションから粒界エネルギーの角度依存性を求め，大槻の実験データとの比較を行った．

角度 θ の値が小さい粒界である小傾角粒界ではうまく実験データを再現出来ているが，角度 θ の値が大きい粒界ではうまく実験データを再現出来ていなかった．本研究で使用した結晶モデル作成プログラムは小傾角粒界の結晶モデルの作成を目的に作られたものである為，角度 θ の大きな粒界の結晶モデルをうまく作成出来ていないことが原因として考えられる．

目 次

第1章　背景

1.1　粒界

物質は単結晶，多結晶，ガラス，非晶質に分けられる．多くの物質は多結晶に属している．多結晶とは図1.1のように単結晶の粒がランダムに隣接しているような状態であり，それぞれの粒には方位がある．方位の違う粒同士が隣接している部分には境界ができ，それを粒界という．

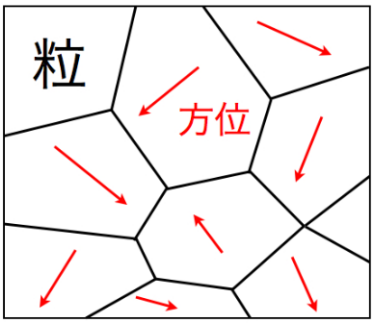


図 1.1: 粒，粒界，結晶方位の模式図

対称傾角粒界とは図 1.2 のように二つの立方晶が回転角 θ 度ずれて結合したものであり，⊥ で表される転位が周期的に並ぶことで構成される．また (100) 面に対称である場合 [100] 対称傾角粒界と呼ばれる．角度 θ が 0 °〜30 °程度のものを小傾角粒界と呼ぶ.

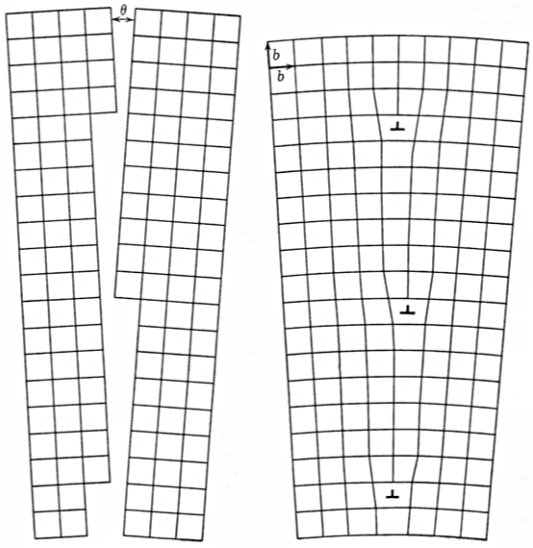


図 1.2: [100] 対称傾角粒界

1.2 粒界エネルギー

粒界が存在しない物体よりも粒界が存在する物体の方がエネルギーが高い状態にあり，その差を粒界エネルギーと呼ぶ．粒界エネルギーは従来，転位論に基づく Read-Shockley の式で求まるとされてきた．この Read-Shockley による予測では角度 0 °でのバーガースベクトルと角度 90 °でのバーガースベクトルが異なるため図 1.3 のように Al[100] 対称傾角粒界のエネルギーの角度 0 °及び 90 °に相当する立ち上がりの傾きが異なることが予測される．ところが近年発表された大槻の実験では図 1.4 のように Al[100] 対称傾角粒界のエネルギーの角度 0 °及び 90 °に相当する立ち上がりの傾きが全く同じであるという矛盾が生じた．この矛盾の解明には，粒界の転位を観察する必要がある．本研究ではこれを粒子シミュレーションから解明することを試みる．

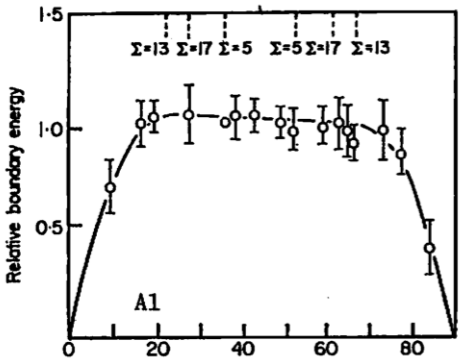


図 1.3: Read-Shockley による粒界エネルギーの角度依存性

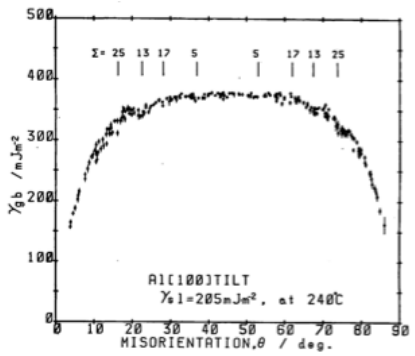


図 1.4: 大槻の実験による粒界エネルギーの角度依存性

第2章　ポテンシャルの検証

粒子シミュレーションで使用する原子間ポテンシャルを検討するため，Lennard-Jones ポテンシャル，EAM ポテンシャル，Morse ポテンシャルの 3 種類のポテンシャルを実装し検証を行った．

2.1　各ポテンシャルについて

Lennard-Jones ポテンシャルは，ばねモデルを数学的に記述する一般的によく使用されているポテンシャルであり式 2.1 で表される．

$$\left.\begin{array}{l} E_i = \sum_i \psi(R_{ij}) \\ \psi(R_{ij}) = A(\frac{1}{R_{ij}})^{12} + B(\frac{1}{R_{ij}})^6 \end{array}\right\} \tag{2.1}$$

Morse ポテンシャルは Lennard-Jones ポテンシャルと距離依存性に少し違いがあるが，本質的には同じものであり，式 2.2 で表される．

$$\left.\begin{array}{l} E_i = \sum_i \psi(R_{ij}) \\ \psi(R_{ij}) = Aexp(-2pR_{ij}) - 2Aexp(-pRij) \end{array}\right\} \tag{2.2}$$

EAM(Embedded atom method) ポテンシャルは，電子の振る舞いを配位数 z の依存性として表現したものであり金属系の物質のシミュレーションに多用されている．式 2.3 で表される．

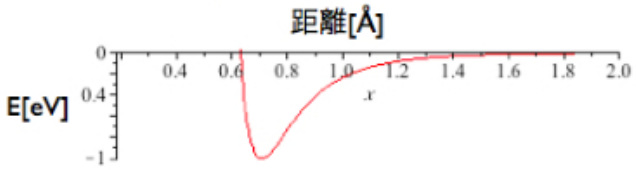
$$E_i = z\psi(R_{ij}) - \sqrt{z}|h(R_{ij})| \tag{2.3}$$

ここで ψ は単純な二体間ポテンシャル，R_{ij} は最近接原子間距離,A,B,p は物性値に合わせて求めた適当な定数である．h は引力を計算するボンド積分を表している．

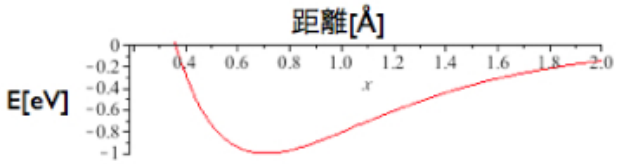
2.2　各原子間ポテンシャルの規格化

平衡原子間距離が $1/\sqrt{2}$[Å] となり，そのときのエネルギー値が -1[eV] となるように任意のパラメータを与える．すると図 2.1 の様な各原子間ポテンシャルの距離依存性が算出される．

- **Lennard-Jones型**



- **Morse型**



- **EAM型**

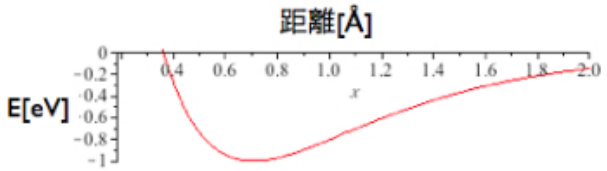


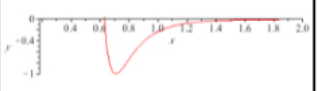
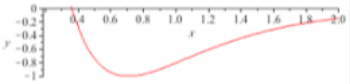
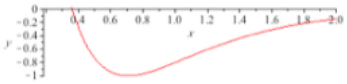
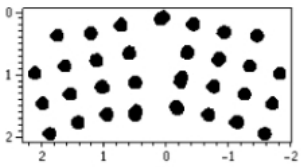
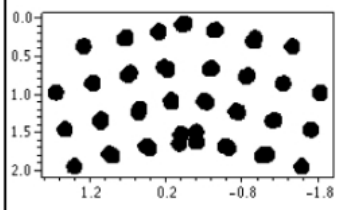
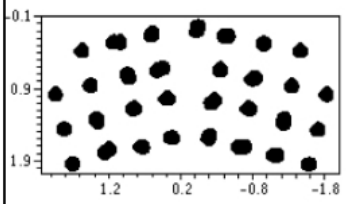
図 2.1: 各原子間ポテンシャルの距離依存性

図 2.1 から Lennard-Jones 型では他の二つに比べて曲率が急になっていて，斥力を大きく計算していることが分かる．

2.3　各ポテンシャルでの計算結果

表 2.1 は各ポテンシャルで計算を行い，その結果をまとめたものである．x-min はエネルギーが最小となる時の x 軸方向への外部緩和を示している．Δ E は粒界結晶の持つエネルギーと完全結晶の持つエネルギーとの差を示している．粒界 E は粒界エネルギーを示している．緩和配置では最安定時の原子配置を示している．表 2.1 の結果から

表 2.1: 各原子間ポテンシャルでの検証結果

	L-J	Morse	EAM
距離依存性			
x-min[Å]	0.30	-0.17	0.13
ΔE[eV]	10.14	-0.72	0.05
粒界E [mJ/m²]	2.43	-0.17	0.01
緩和配置			

- Morse ポテンシャルと EAM ポテンシャルでは距離依存性の形がかなり似ているのにも関わらず得られた粒界エネルギーが大きく違う．
- Morse ポテンシャルでは粒界結晶の持つエネルギーが完全結晶のもつエネルギーよりも低い値となった．
- Lennard-Jones ポテンシャルと EAM ポテンシャルでは最安定時の原子位置はかなり似ているのに対して得られた粒界エネルギーが大きく違う．

などが読み取れる．
表 2.1 の結果が得られた原因として，Morse ポテンシャルでは斥力が小さく計算されるため，原子同士を近づけるとボンドの数を増やしてより安定な値になるという性質があり，近づける程安定であるという結果が得られる．EAM ポテンシャルでは結合がひとつ減ったときに電子が空孔とは反対の結合側に流れ，その結合を強化するバックボンド現象を表現しているため他のポテンシャルとは違った値が得られたと考えられる．この考察から 3 つのポテンシャルの中で EAM ポテンシャルが本研究を進める上で最も適していると考え，EAM ポテンシャルを用いて計算を続けていく．

第3章　手法

3.1　計算の流れ

検証の手法として図3.1の流れで計算を行うプログラムを作成した．以後の節で各処理についての説明を行う．

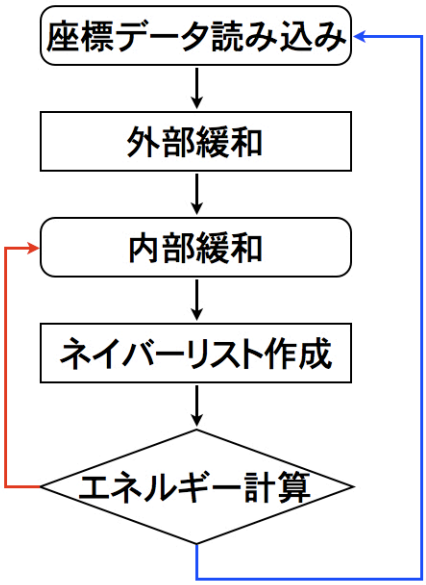


図 3.1: 計算のフローチャート

3.2　座標データの読み込み

以下の関数で初期座標データを読み込み，配列 lattice として帰すという作業を行っている．

```
int read_lattice(void){
```

```
  int i=0;
  FILE *fp;
  fp=fopen("初期座標データ","r");
  fscanf(fp,"%d %lf %lf %lf %lf",&n_max, &LL[0], &LL[1], &LL[2], &theta);
  LL[1]=LL[1]/cos(theta);
   for(i=0;i<n_max;i++){
    fscanf(fp,"%lf %lf %lf",&lattice[i][0],&lattice[i][1],&lattice[i][2]);
   }
}
```

初期座標データの中身は図 3.2 のようになっている．1 行目は原子数，x 軸方向の大きさ,y 軸方向の大きさ,z 軸方向の大きさ, 回転角 θ の値を左から順に記述している．ここで，x 軸方向の大きさが 100 となっているのは以降の節で説明する固定境界条件の為である．二行目以降は左から各原子の x 座標,y 座標,z 座標を記述している．

```
126 100 4 2 0.1243549945
0.000000 0.000000 0.000000
1.922538 0.744208 0.500000
0.496139 0.062017 0.500000
0.434122 0.558156 0.000000
0.000000 0.000000 1.000000
1.922538 0.744208 1.500000
0.496139 0.062017 1.500000
0.434122 0.558156 1.000000
1.860521 1.240347 0.000000
```

図 3.2: 初期座標データの例

図 3.2 に表される初期座標を maple を使いプロットさせ,z 軸方向から見たものが図 3.3 である．

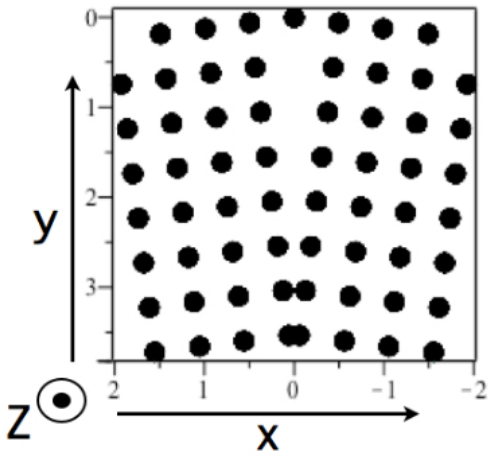


図 3.3: 初期座標のプロット

3.3 境界条件

3.3.1 周期的境界条件

結晶モデルのエネルギー計算は隣接する結晶モデルと連続した状態であることを仮定して行われる．そのため，以下に示したように原子間距離を求める関数に周期的境界条件の式を実装する必要がある．

```
double dist(double a[3], double b[3]){
  double dist1=0.0, dd=0.0;
  int i;
  for (i=0;i<3;i++){
    dd=a[i]-b[i];
    dd=dd-round(dd/LL[i])*LL[i];//周期的境界条件の式
    dist1 += dd*dd;
  }
    return sqrt(dist1);
}
```

周期的境界条件を実装すると，図3.4のように主体となるセルの端に原子が当たると当たった原子が反対側から出てくるような状況を作ることが出来る．そのため，1つの結晶モデルを使って同じセルが周囲に無数に並列しているモデルの計算を行うことが出来るようになる．

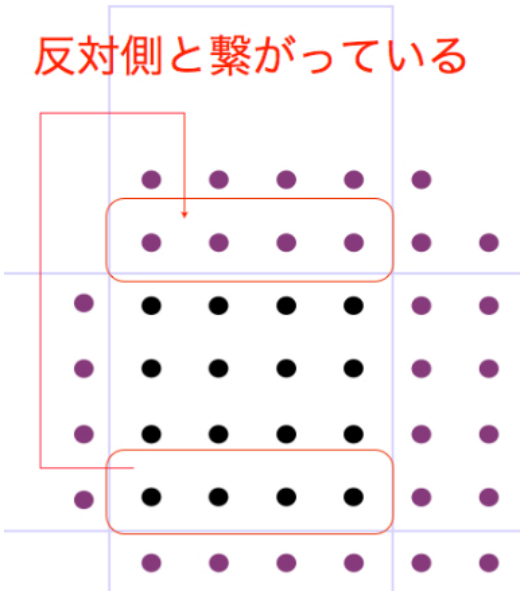


図 3.4: 周期的境界条件

3.3.2 固定境界条件

x 軸方向に対しては周期的境界条件を適用せず，固定境界条件を適用する．固定境界条件とは未知数の部分を既知の値に固定する手法である．ここでは図 3.5 に示されるように x 軸方向の両端の一番外の原子が持つエネルギー値を計算せず，-1 に固定する．完全結晶状態の持つエネルギーは-1 であり，粒界結晶の粒界面から遠い位置にある構造は完全結晶と殆ど同じものであるからである．

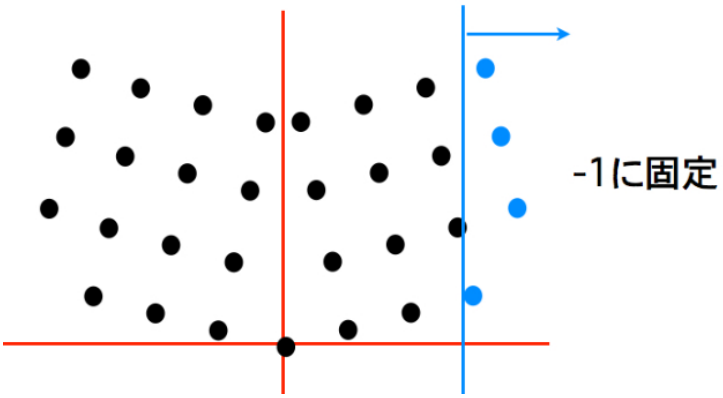


図 3.5: 固定境界条件

青色の縦線よりも内側にある原子は外側の原子とも繋がった状態でエネルギー値を

計算することが出来るが，外側にある原子のエネルギー値は-1 に固定される．エネルギー値が-1 に固定される原子以外を配列 calc_atom として取得する関数を以下に示す．ここでは一番外側の原子座標を max_x,min_x として指定し，その位置から 0.51*cos θ 以内の原子を除去している．

```
int mk_calc_atom(void){
  int h,i,i_atom=0;
  double x_cut=0.51;
  double max_x=-100.0,min_x=100.0;
  for(h=0;h<n_max;h++){
    if (lattice[h][0]>max_x){
      max_x=lattice[h][0];
    }
    else if (lattice[h][0]<min_x){
      min_x=lattice[h][0];
    }
  }
  for(i=0;i<1000;i++){ calc_atom[i]=-1; }
  for(i=0;i<n_max;i++){
    if ((lattice[i][0]<max_x-x_cut*cos(theta))&&(lattice[i][0]>min_x+x_cut*cos(the
      calc_atom[i_atom++]=i;
    }
  }
}
```

3.4 外部緩和

外部緩和とは，図 3.6 のように結晶モデルの界面よりも右側をブロック単位で動かすことで原子の安定位置を模索する手法である．以下の関数で界面よりも右側の原子位置を移動している．

```
int blockmove(double dx,double dy,double dz){
  int out;
  for(out=0;out<n_max;out++){
    if(lattice[out][0]>0.0){
      lattice[out][0]=lattice[out][0]+dx;
      lattice[out][1]=lattice[out][1]+dy;
      lattice[out][2]=lattice[out][2]+dz;
    }
```

```
  }
}
```

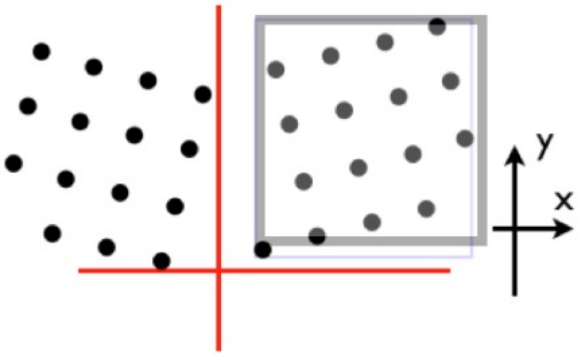


図 3.6: 外部緩和

3.5 内部緩和

内部緩和とは，図 3.7 のように結晶モデル内の原子を一つずつバラバラに動かして最安定位置を求める手法である．

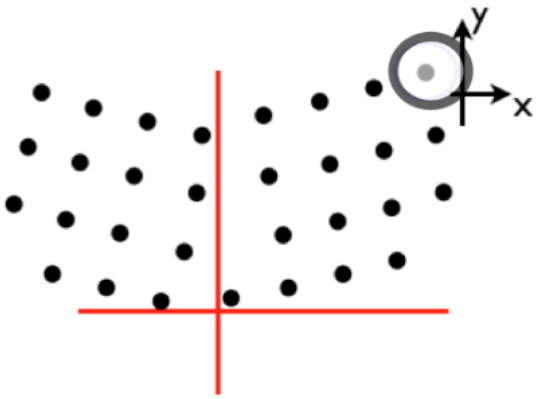


図 3.7: 内部緩和

内部緩和はモンテカルロ法を採用し，以下の関数で計算する．

```
double innerRelax(int mc){
  int h,i=0,j,atom;
  double de,dd;
```

```
    double T=0.001;
  srand((unsigned) time(NULL));
  for(h=0;h<mc;h++){
    reset_NL();
    mk_NL();
    atom=0,i=0;
    while((atom=calc_atom[i++])>=0){
      for(j=0;j<3;j++){
dd=(1.0-2.0*(rand()%10000/10000.0))/100.0;
de=e_diff(atom,dd,j);
if(exp(de/T)<(rand()%10000/10000.0)){//モンテカルロ法
  lattice[atom][j]-=dd;
}
else{
}
      }
    }
  }
}
```

3.6 ネイバーリスト

ネイバーリスト (以下 NL) は近接している原子の集合を意味する．エネルギー値を計算するにあたり全ての原子間の相互作用を考慮すると計算に時間がかかりすぎるので，影響力の弱い，原子から遠い位置にある原子との相互作用を遮断する．図 3.8 のように赤色で示す 1 個の原子から青色で示す既定の範囲内にある原子を NL に登録し，NL に登録した原子との間の相互作用エネルギーの総和が 1 つの原子の持つエネルギーであると考える．ここでは最近接原子よりも 1 つ奥にある原子の手前までを NL 範囲に指定することで最近接原子のみを NL に登録する．これらの作業を以下の関数によって行う．

```
int mk_NL(void){
  int i,j,i_nl,i_atom;
  double d1;
  j=0;
  while((i_atom=calc_atom[j++])>=0){
    i_nl=0;
    for(i=0;i<n_max;i++){
      d1=dist(lattice[i_atom],lattice[i]);
```

```
      if (i!=i_atom){
if ((d1<0.9)){//最近接原子のみを NL に登録
  nl[i_atom][i_nl]=i;
  i_nl++;
}
      }
    }
  }
}
```

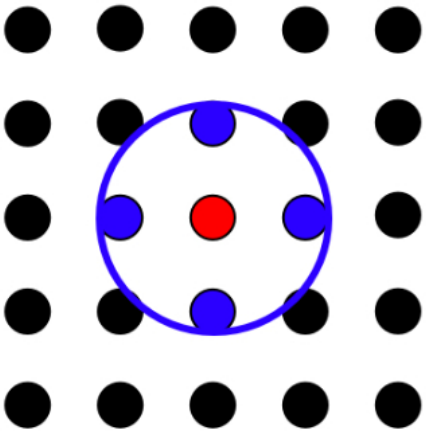


図 3.8: ネイバーリスト

3.7 粒界エネルギーの計算

前章で説明した様に，原子同士の相互作用エネルギーの計算に以下の関数で示される EAM ポテンシャル (Embedded atom method) を使用した．

```
double eam_energy(int i_atom){
  double q,a0,b0,poq,p,rep,rho,x0,E0;
  int i,i_nl;
  q=4.00;a0=23.8538969400;b0=9.7680902790;
  poq=2.0;p=q*poq;

  rep=rho=0.0;
  E0=0.0;
  i_nl=0;
```

```
  for (i_nl=0;nl[i_atom][i_nl]>=0;i_nl++) {
    i=nl[i_atom][i_nl];
    x0=dist(lattice[i_atom],lattice[i]);
    rep+=exp(-p*x0);
    rho+=exp(-2.0*q*x0);
    E0=a0*rep-b0*sqrt(rho);
   }
  return E0;
}
```

結晶モデル内にある原子の持つエネルギーを上記の関数で求め，それらの総和 e_total を以下の関数で求める．

```
double e_total(void){
  int i,j=0,i_atom;
  double e_total=0.0;
  while((i_atom=calc_atom[j++])>=0){
       e_total+=eam_energy(i_atom);
   }
    return e_total;
}
```

粒界エネルギーは，完全結晶の持つエネルギー値と粒界結晶の持つエネルギー値の差を単位面積辺りに換算したものである．完全結晶は 1 つの原子の持つエネルギーが− 1 なので，完全結晶の e_total は [-原子の個数] と同じ値になる．粒界結晶のエネルギー値は上記の関数で求めた e_total を使用する．

$$\text{粒界エネルギー} = \frac{\text{粒界結晶の } e\_total - \text{完全結晶の } e\_total}{\text{面積}} \tag{3.1}$$

第4章　RubyとC言語との計算時間検証

これまで，プログラミング言語 Ruby を使用して検証を行っていたが，計算時間が掛かり過ぎるという問題が生じた．そこで，より計算時間が早いとされる C 言語を使用することによって計算時間の短縮を試みた．計算精度を表すモンテカルロの試行回数を 500 回にして比較を行った．その結果，表 4.1 が得られた．

表 4.1: Ruby と C 言語の計算時間の比較

	Ruby	C言語
MC500回での計算時間(s)	214	8

表 4.1 から分かる様に Ruby から C 言語に書き換える事によって計算時間が大幅に短縮出来た．これにより，更に大きなモデルでの検証や様々なパラメータでの検証が可能となった．Ruby,C 言語の二つのコードでの検証が可能となり，実際の計算には計算時間の早い C 言語を使用し，計算エラーの対応などにはコンパイル作業が不要なインタープリタ型の言語である Ruby を使用することが可能となった．

第5章　EAMポテンシャルのパラメータの検証

EAM ポテンシャルでのエネルギー計算は式 5.1 で行う．

$$\left.\begin{aligned} energy &= z * rep - \sqrt{B * z * hop^2} \\ rep &= A * exp(-p * r) \\ hop &= exp(-q * r) \end{aligned}\right\} \tag{5.1}$$

ここで z は配位数，p，q は距離依存性の曲率を決めるパラメータである．rep は斥力を，hop は引力をそれぞれ表している．

5.1　パラメータ poq

これまでは p や q の値は単に距離依存性の曲率を決定するパラメータであり，シミュレーションの結果に大きな影響は無いと考え z=12,q=2.0,poq=2.0,p=q*poq と適当な値を代入し，計算を行っていた．この時の距離依存性は図 5.1 の様になる．

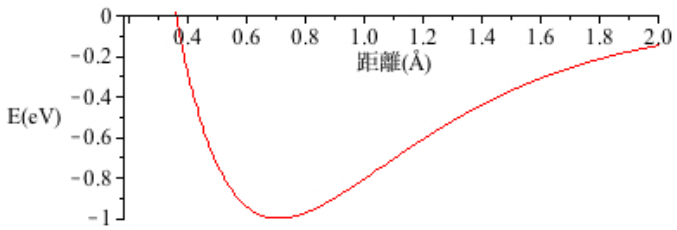


図 5.1: パラメータ q=2.0,poq=2,0 のときの EAM ポテンシャルの距離依存性

このパラメータを使用してモンテカルロ法，最急降下法で計算を行うと図 5.2，図 5.3 の結果が得られた．

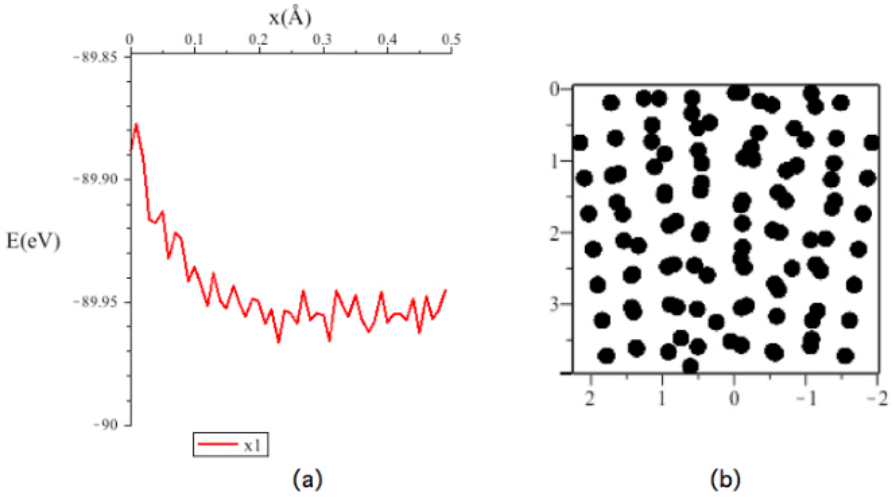


図 5.2: パラメータ q=2.0,poq=2.0 のときのモンテカルロ法での結果.(a)x 軸方向への外部緩和とエネルギー値との相関，(b) 最安定時の原子配置.

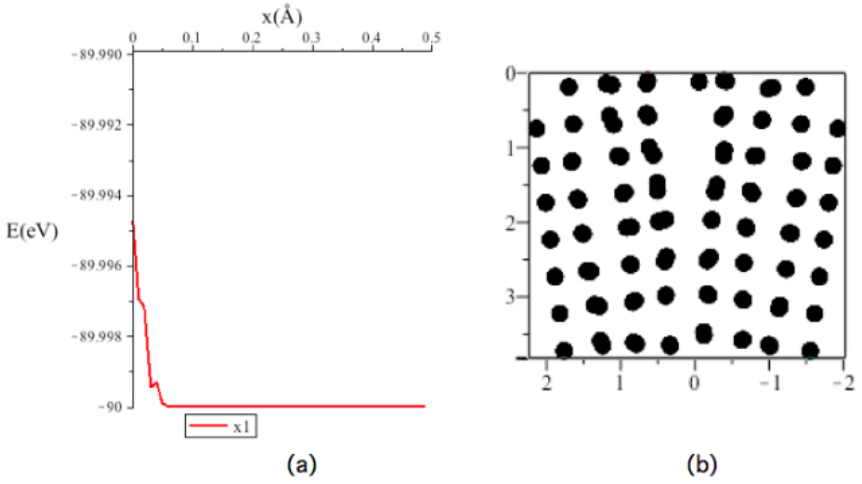


図 5.3: パラメータ q=2.0,poq=2.0 のときの最急降下法での結果.(a)x 軸方向への外部緩和とエネルギー値との相関，(b) 最安定時の原子配置.

モンテカルロ法での検証の結果では図 5.2 の (a) の様にエネルギー値は上下に波打ちながら下がっていき，(b) の様に最安定時の原子配置がバラバラになってしまっている．最急降下法での検証の結果では図 5.3 の (a) の様にエネルギーが-90 に安定している事が分かる．この時の原子数が 90 で完全結晶のエネルギーが− 1eV で計算を行っているので界面のエネルギーが完全結晶のエネルギーと全く同じであるという意図しない結果が得られた．最急降下法での計算過程を詳しく見てみると，配位数が 12 以外の場合でも-1eV のエネルギー値を算出していることが分かった．そこで EAM ポテンシャルの配位数別の距離依存性を算出し比較すると図 5.4 の結果が得られた．

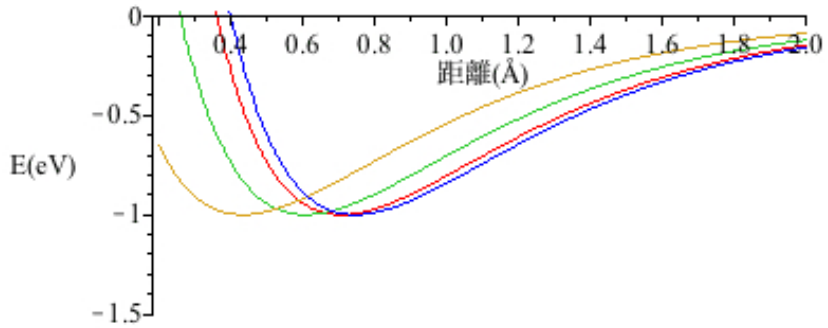


図 5.4: パラメータ poq=2.0 のときの EAM の配位数別の距離依存性．黄:配位数 4, 緑:配位数 8，赤:配位数 12，青:配位数 14

図 5.4 から poq=2.0 の場合どんな配位数でも最安定のエネルギー値が-1eV と算出されてしまうことが分かった．そこで poq の値を 2.0 から 3.0 に変更して配位数別の距離依存性を算出し比較すると図 5.5 の結果が得られた．

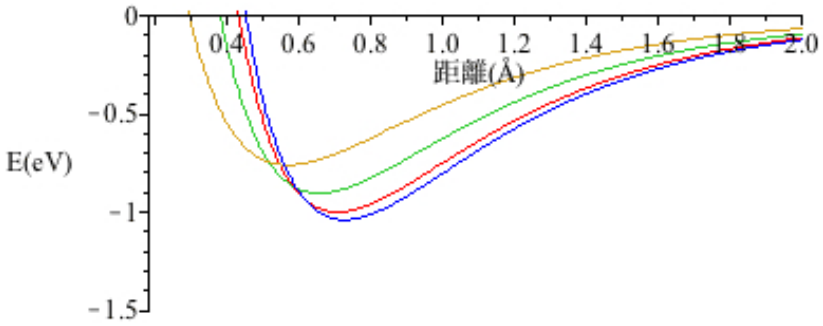


図 5.5: パラメータ poq=3.0 のときの EAM の配位数別の距離依存性．黄:配位数 4, 緑:配位数 8，赤:配位数 12，青:配位数 14

図 5.5 からパラメータ poq の値が 3.0 の場合は配位数によって最安定のエネルギー値が違う距離依存性が得られていることが分かる．図 5.4 と図 5.5 の結果から

パラメータ poq の値は 2.0 以外の数で計算を行う必要があると分かった．パラメータ poq の値を 3.0 にして計算を行うと図 5.6 と図 5.7 の結果が得られた．

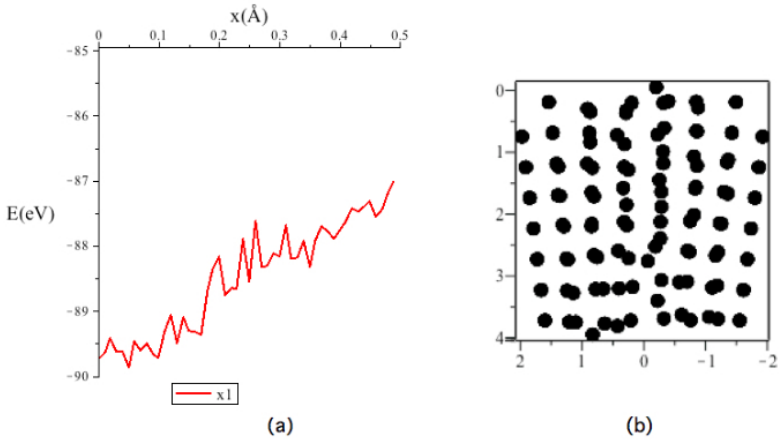


図 5.6: パラメータ q=2.0,poq=3.0 のときのモンテカルロ法での結果.(a)x 軸方向への外部緩和とエネルギー値との相関，(b) 最安定時の原子配置．

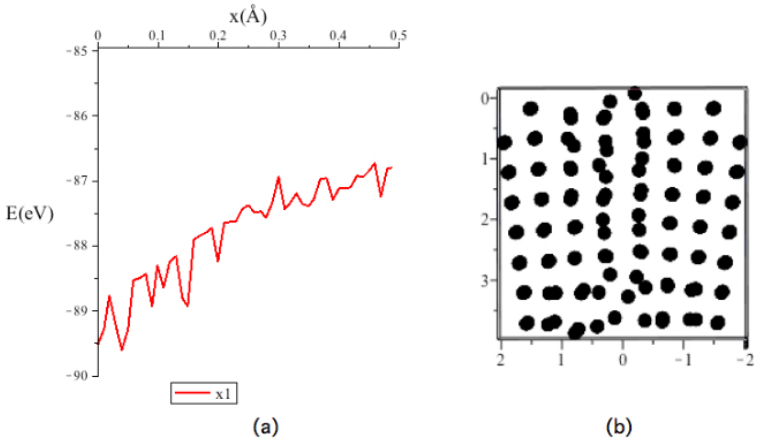
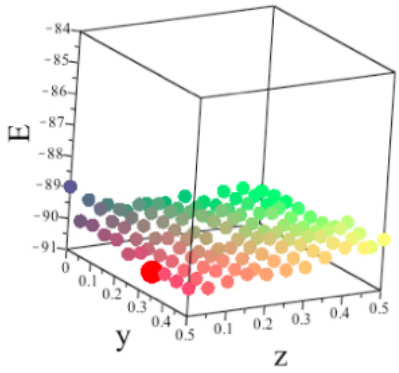


図 5.7: パラメータ q=2.0,poq=3.0 のときの最急降下法での結果.(a)x 軸方向への外部緩和とエネルギー値との相関，(b) 最安定時の原子配置．

図 5.6 と図 5.7 の結果からパラメータ poq の値を 3.0 にすると最安定時の原子配置がバラバラになることが無く，界面エネルギーが-90 で安定することも無いのでパラメータ poq の値が 2.0 の時と比べて, より意図する結果が得られていることが分かる．これによりパラメータ poq の値は 2.0 以外の数で行う必要があると分かった．

5.2　パラメータ q

これまでは外部緩和を x 軸方向にのみ行っていたが，y 軸方向と z 軸方向への外部緩和も加えて計算を行った．すると図 5.8 の結果が得られた．



赤い点が最安定
(-90.21111eV)を示す.

図 5.8: パラメータ q=2.0 のときの y 軸，z 軸，エネルギーの最小値の 3 次元プロット．

図 5.8 のようにパラメータ q が 2.0 のとき 3 方向へ外部緩和を行うと最安定エネルギーが-90.21111eV となり完全結晶よりも安定であるという意図しない結果が得られた．原因として図 5.5 の青い線で示している配位数 14 で安定していまっていることが予測される．そこで図 5.9 のようにパラメータ q を 2.0 から 3.0 に変更し反発力を大きくして検証を行った．すると図 5.10 の結果が得られた．

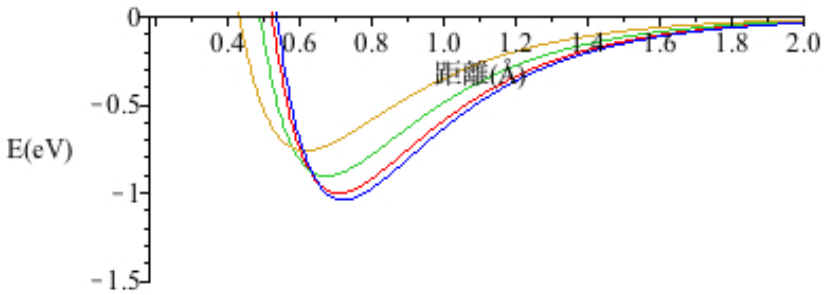
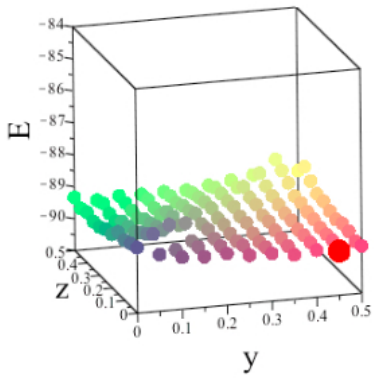
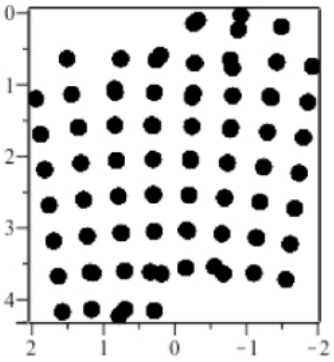


図 5.9: パラメータ q=3.0 のときの距離依存性



x,y,z方向に外部緩和させ
plot(E_min,y,z)させた.



エネルギーが最安定の時の原子配置.
このときの外部緩和
x軸方向:0.02Å
y軸方向:0.45Å
z軸方向:0.00Å
エネルギー:-89.61786eV

図 5.10: パラメータ q=3.0 のときの y 軸，z 軸，エネルギーの最小値の 3 次元プロットと最安定時の原子配置.

図 5.10 からパラメータ q の値を 3.0 にすることによって反発力が大きくなり，配位数 12 の時の方が配位数 14 の時よりも安定となり, より意図する結果が得られていることが分かる．今後はパラメータ poq=3.0,q=3.0 を使用して検証を進めていく.

第6章　小傾角粒界エネルギーの角度依存性

これまではθ=arctan(1/8）の座標データのみを使用し，検証を進めて来た．これに加えて更にθ=arctan(1/2),arctan(1/4),arctan(1/16)でも同様の計算を行い粒界エネルギーの角度依存性の検証を行った．ここでのθの値は図6.1のθを指すので角度依存性で使用する角度は2 θとなる．また各計算で使用する初期座標データは角度θに合わせて拡張する必要がある．以下の節で各角度での計算結果，得られた粒界エネルギーから求めた小傾角粒界エネルギーの角度依存性を解説する．

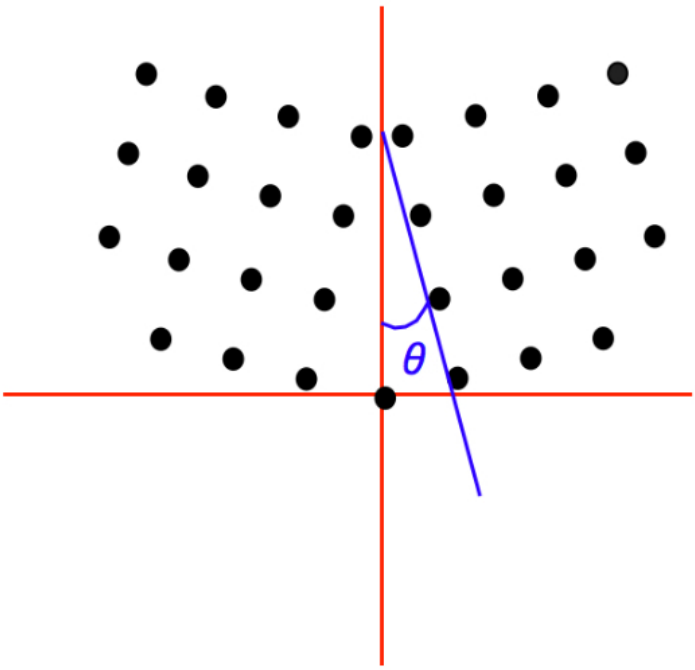


図 6.1: θの模式図.

6.1 arctan(1/16)

θ =arctan(1/16) で検証を行った．図 6.2 は θ =arctan(1/16) での初期座標と緩和後の原子配置である．

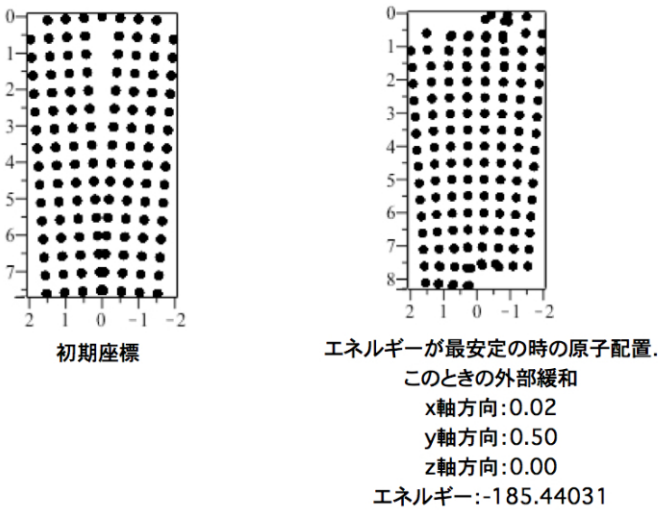


図 6.2: θ =arctan(1/16) での初期座標と緩和後の原子配置．

図 6.3 は外部緩和とエネルギー値との相関である．

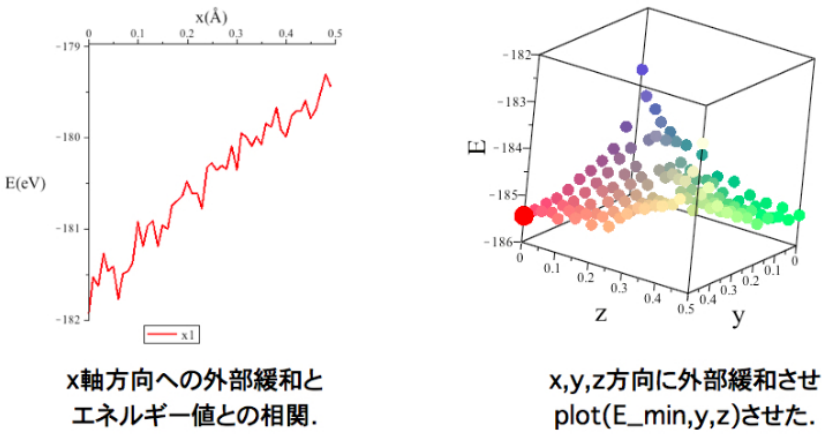
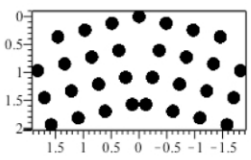


図 6.3: θ =arctan(1/16) での x 軸，エネルギーの最小値の 2 次元表示と y 軸，z 軸，エネルギーの最小値の 3 次元表示．

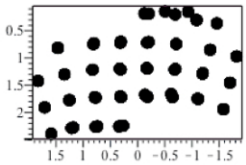
特徴として，図 6.2 から y 軸方向に大きく外部緩和を行ったときに安定になっていることが分かる．図 6.3 の 2 次元表示からは θ = arctan(1/8) の時と同様に上下に波打ちながら上昇していることが分かる．

6.2 arctan(1/4)

θ =arctan(1/4) で検証を行った．図 6.4 は θ =arctan(1/4) での初期座標と緩和後の原子配置である．



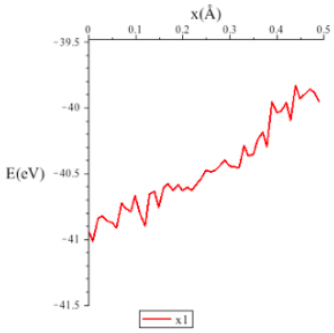
初期座標



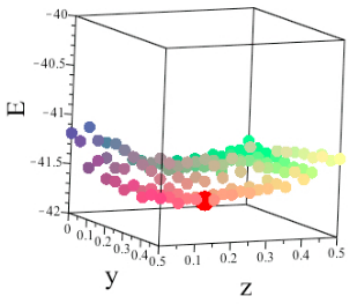
エネルギーが最安定の時の原子配置．
このときの外部緩和
x軸方向:0.01
y軸方向:0.45
z軸方向:0.15
エネルギー:-41.60363

図 6.4: θ =arctan(1/4) での初期座標と緩和後の原子配置．

図 6.5 は外部緩和とエネルギー値との相関である．



x軸方向への外部緩和と
エネルギー値との相関．



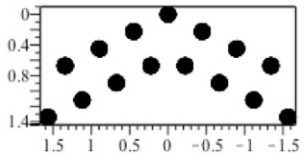
x,y,z方向に外部緩和させ
plot(E_min,y,z)させた．

図 6.5: θ =arctan(1/4) での x 軸，エネルギーの最小値の 2 次元表示と y 軸，z 軸，エネルギーの最小値の 3 次元表示．

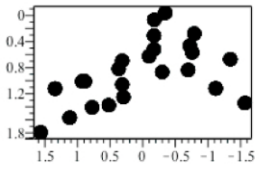
特徴として，図 6.4 から y 軸方向に大きく外部緩和を行ったときに安定になっていることが分かる．図 6.5 の 2 次元表示からは θ = arctan(1/8) の時と同様に上下に波打ちながら上昇していることが分かる．

6.3 arctan(1/2)

θ =arctan(1/2) で検証を行った．図 6.6 は θ =arctan(1/2) での初期座標と緩和後の原子配置である．



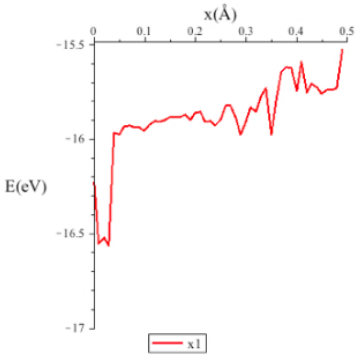
初期座標



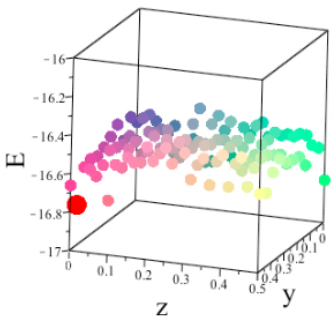
エネルギーが最安定の時の原子配置．
このときの外部緩和
x軸方向:0.00
y軸方向:0.45
z軸方向:0.00
エネルギー:--16.78916

図 6.6: θ =arctan(1/2) での初期座標と緩和後の原子配置．

図 6.7 は外部緩和とエネルギー値との相関である．



x軸方向への外部緩和と
エネルギー値との相関．



x,y,z方向に外部緩和させ
plot(E_min,y,z)させた．

図 6.7: θ =arctan(1/2) での x 軸，エネルギーの最小値の 2 次元表示と y 軸，z 軸，エネルギーの最小値の 3 次元表示．

図 6.6 を見ると緩和後の原子配置がバラバラになってしまい，うまく緩和出来ていないことが分かる．また，図 6.7 の 2 次元表示からは他の角度のときと違った形が算出された．

6.4　角度依存性

それぞれの角度で得られた粒界エネルギーから粒界エネルギーの角度依存性を検証した．規格化した距離依存性を使用して得られた結果に Al の物性値をかけ，大槻の実験結果と同様に横軸に角度，縦軸に粒界エネルギーをとると図 6.8 が得られた．

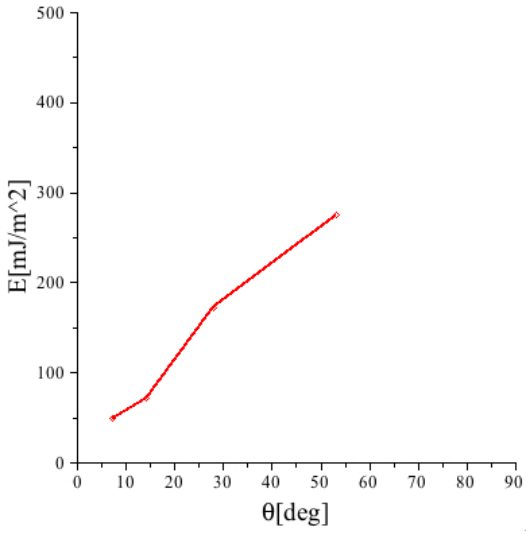


図 6.8: シミュレーションから求めた小傾角粒界エネルギーの角度依存性.

更に大槻の実験結果との比較のため図 6.9 のようにシミュレーションの結果と大槻の実験結果とを重ねて表示させた．

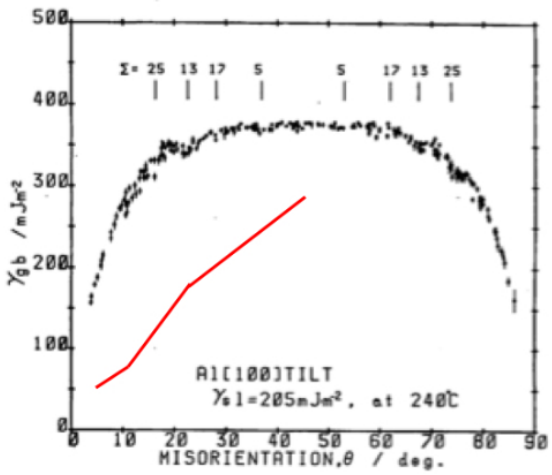


図 6.9: シミュレーションでの結果と大槻の実験結果.

図 6.9 から粒界エネルギーの絶対値に若干ズレは有るものの 20 °から 30 °付近での立ち上がりを見ると実験結果と良く似た傾きが得られている．つまり角度 θ が 30 °以内の小傾角粒界では実験値を良く再現出来ている．しかし，角度 θ が 30 °を超える粒界では実験結果と異なる傾きが算出されており，実験値をうまく再現出来ていない．本研究で使用した結晶モデル作成プログラムでは角度 θ の値が大きい結晶モデルをうまく作成出来ないことが示唆される．今後，粒子シミュレーションでの角度依存性と大槻の実験結果とのズレを解消する為，更なる検証が必要である．

第7章　総括

本研究では，大槻が Al の粒界に関する実験から示唆した矛盾を解消することを目的に粒子シミュレーションを行った．

本シミュレーションに使用するポテンシャルを決めるため，Lennard-Jones ポテンシャル，Morse ポテンシャル，EAM ポテンシャルの 3 種類で検証を行った．その結果，それぞれのポテンシャルで異なる結果が得られることが分かり，その中でもバックボンド強化を再現する EAM ポテンシャルが本研究において最適であると分かった．

シミュレーションに使用する EAM ポテンシャルの式

$$\left.\begin{aligned} energy &= z * rep - \sqrt{B * z * hop^2} \\ rep &= A * exp(-p * r) \\ hop &= exp(-q * r) \end{aligned}\right\}$$

について様々なパラメータの値に対して，得られる結果を調べた．その結果，パラメータ p/q は 2．0 にすると，どの配位数でも最安定エネルギーが同じになってしまうため，2.0 以外の値で計算する必要があると分かった．更に，パラメータ q は低い値に設定すると斥力が小さくなりすぎて配位数が多いほど安定になってしまうので，3.0 程度で計算を行う必要があると分かった．そして実際に得られたパラメータを用いてシミュレーションを行った．

シミュレーションによって得られた結果から粒界エネルギーの角度依存性を求め，大槻の実験データとの比較を行った．すると 30 °以内の小さい角度の粒界である小傾角粒界では実験データの傾きと良く似た結果が得られ，実験データをうまく再現出来た．しかし，30 °以上の大きい角度の粒界では実験データの傾きとは異なった結果が得られ，実験データをうまく再現出来ていないことが分かった．本研究に使用した結晶モデル作成プログラムは小傾角粒界の結晶モデルの作成を目的に作られたものであり，角度 θ が大きい粒界の結晶モデルをうまく作成出来ていないことが再現出来ない原因として考えられる．まだ，小傾角粒界でもシミュレーションによる角度依存性と大槻の実験結果とでズレがある為，今後更なる検証が必要である．

また，計算プログラムを Ruby から C 言語に書き換えることによって計算時間を約 1/25 に短縮することに成功した．

参考文献

[1] 鈴木秀次,「転位論入門」,(アグネ　1967).

[2] 大槻徴,「アルミニウムの粒界エネルギーに関する研究」,(京都大学工学研究科博士論文　1990).

[3] 岩倉新太郎,「Lennard-Jones ポテンシャルによる粒界エネルギーの計算」,(関西学院大学情報科学科卒業論文　2011).

[4] 西谷滋人,「固体物理の基礎」,(森北出版株式会社　2006).

謝辞

本研究の遂行にあたり，終始多大な有益な御指導，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，西谷研究室に所属する同輩達, 並びに先輩方からの様々な知識の供給，御協力を頂き，本研究を成就させる事ができました．この場を借りて心から深く御礼申し上げます．

付 録A　rubyによる粒界エネルギー算出コード

```
include Math
require 'pp'

$m = 1.000

class Atom
  attr_accessor :pos, :pos0, :nl
  def initialize(pos)
    @nl=[]
    @pos=Array.new(pos)
    @pos0=Array.new(pos)
  end
  def energy()
#   return lj_energy()
#   return morse_energy()
   return eam_energy()
  end

  def reset_nl()
    @nl=[]
  end

  def lj_energy()#LJP
    a0=1.587401051
    e0=-1*4.0/12.0

    ene=0.0
    nl.each do |j|
      r=distance(@pos,$atom_list[j].pos)
      a=1.0/(a0*r)
```

```
      ene+=e0*((a**6)-(a**12))
    end
    return ene
  end


  def morse_energy()#Morse potential
    q=2.0;a0=1.409902389;b0=0.6855417295
#    q=4.00;a0=23.8538969500;b0=2.8198047780;
    poq=2.0;p=q*poq
    ene=0.0
    nl.each do |j|
      r=distance(@pos,$atom_list[j].pos)
      ei=a0*exp(-p*r)-b0*exp(-q*r)
      ene+=ei
    end
    return ene
  end

  def eam_energy()#EAM potential
#    q=2.00;a0=1.4099023880;b0=2.3747862110;
    q=4.00;a0=23.8538969400;b0=9.7680902790;
    poq=2.0;p=q*poq

    rep=rho=0.0
    nl.each do |j|
      x0=distance(@pos,$atom_list[j].pos)
      rep+=exp(-p*x0)
      rho+=exp(-2.0*q*x0)
    end
    return a0*rep-b0*sqrt(rho)
  end
end

def makeLattice()
  file = open('2428ruby.txt')
  atom_list=[]
  while line = file.gets do
    pos0=line.chomp.split(" ")
```

```
    pos1=[]
    pos0.each do |p|
      pos1 << p.to_f
    end
    atom_list << Atom.new(pos1)
  end
  file.close
#puts atom_list.size
#exit
  return atom_list
end

$nx=2*$m
$ny=4*$m
$nz=2*$m
$theta=atan(1.0/8.0)
$L=[100.0,$ny/cos($theta),$nz]
def distance(a,b)
  tmp=0
  for i in 0..2 do
    x=a[i]-b[i]
    x=x-(x/$L[i]).round*$L[i]
    tmp+=x*x
  end
  return  sqrt(tmp)
end

def distancetes(a,b)
  tmp=0
  pp a
  pp b
  for i in 0..2 do
    x=a[i]-b[i]
    x=x-(x/$L[i]).round*$L[i]
    tmp+=x*x
  end
  return  sqrt(tmp)
end
```

```
def mkAtom_include
  atom_include=[]
  max_x = -100
  min_x = 100
  $atom_list.each do |i_a|
    if i_a.pos[0] > max_x then
      max_x = i_a.pos[0]
    elsif i_a.pos[0] < min_x then
      min_x = i_a.pos[0]
    end
  end

  $atom_list.each_with_index do |i_a,idx|
    if i_a.pos[0] < max_x-0.51*$m*cos($theta) && i_a.pos[0] > min_x+0.51*$m*cos($t
      atom_include << idx
    end
  end
  return atom_include
end

def makeNL()
  $atom_list.each do |ai| ai.reset_nl end
  nmax=$atom_list.length-1
  for i in 0..nmax do
    ai=$atom_list[i]
    for j in i+1..nmax do
      aj=$atom_list[j]
      if distance(ai.pos,aj.pos)<0.9*$m then
        ai.nl << j
        aj.nl << i
      end
    end
  end
#puts aj.nl.size
end

def block_move(dx,dy,dz)
  $atom_list.each do |atom|
```

```
    if atom.pos0[0]>0.0 then
      atom.pos[0]=atom.pos0[0]+dx
      atom.pos[1]=atom.pos0[1]+dy
      atom.pos[2]=atom.pos0[2]+dz
    end
  end
end

def total_E()
  total_E=0.0
  $atom_include.each do |i|
    total_E+=$atom_list[i].energy()
  end
  return total_E
end

def diff_E(atom,j,dx)
  e0 = atom.energy()
  atom.nl.each do |i|
    if $atom_include.member?(i) then
      e0+= $atom_list[i].energy()
    end
  end

  atom.pos[j]+=dx
  e1 = atom.energy()
  atom.nl.each do |i|
    if $atom_include.member?(i) then
      e1+= $atom_list[i].energy()
    end
  end
#puts "e0",e0
#puts "e1",e1
  diff_E=e0-e1
  atom.pos[j]-=dx
  return diff_E
end

T=0.001
```

```
$MC=false
def InnerRelax(trial)
  trial.times do
    makeNL()
#pp $atom_list
    $atom_include.each do |i|
      atom=$atom_list[i]
      #pp atom
      for j in 0..2 do
        if $MC then
          $trial+=1;$accept+=1
          e0=total_E()
          dx=(1.0-2.0*rand())/100.0
          de=diff_E(atom,j,-dx)
          if exp(-de/T)<rand() then #mc
           $accept-=1
           atom.pos[j]-=dx
         end
        else
          $trial+=1;$accept+=1
          dx=(1.0-2.0*rand())/100.0
          atom.pos[j]+=dx
          de=diff_E(atom,j,-dx)
          if de>0 then
            $accept-=1
            atom.pos[j]-=dx
          else
#            puts"YEAH"
          end
        end
      end
    end
   #printf("accept, trial:%d %d \n",$accept,$trial)
  end
end

def brent(x0,x1,x2,xmin)
  block_move(x0,dy,dz)
  e0=total_E()
```

```
  return emin
end

def OuterRelax(nrange)
  rd1=0.0
  min=[0.0,0.0,0.0,1000]
  for zz in 0..nrange do
    for yy in 0..nrange do
      for xx in 0..10 do
        $trial=0
        $accept=0
        $atom_list=makeLattice()
        $atom_include=mkAtom_include()

        dx=xx/20.0
        dy=yy/20.0
        dz=zz/20.0
        #dx=(xx/100.0)
        #dy=0
        #dz=0
        block_move(dx,dy,dz)
        makeNL()
        InnerRelax(0)
        e0=total_E()
        printf("%10.5f %10.5f %10.5f %10.5f\n",dx,dy,dz,e0)
        if min[3]>e0 then
          min=[dx,dy,dz,e0]
        end
      end
    end
  end
  printf("%10.5f %10.5f %10.5f %10.5f\n",min[0],min[1],min[2],min[3])

  $atom_list=makeLattice()
  $atom_include=mkAtom_include()

  boundary_E=(min[3]-(-$atom_include.size))/($ny*(1/cos($theta))*$nz)
  printf("boundary_E\%10.5f \n",boundary_E)
```

```
  dx=min[0]
  dy=min[1]
  dz=min[2]

  block_move(dx,dy,dz)
  makeNL()
  InnerRelax(0)

end

$t0=Time.now
nrange=10
OuterRelax(nrange)

file = File.open(ARGV[0],'w')
for i in 0..$atom_list.size-1
  file.printf("%10.5f %10.5f %10.5f \n",$atom_list[i].pos[0],$atom_list[i].pos[1],$
  end
file.close
puts $atom_include.size
p Time.now-$t0
```

付 録B　C言語による粒界エネルギー算出コード

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
int n_max;
double lattice[5000][3];
double result_lattice[5000][3];
int nl[5000][20];
int calc_atom[5000];

double theta;
double LL[3];
double x_cut=0.51;
//double x_cut=2.01;
double T=0.001;

int read_lattice(void){
  int i=0;
  FILE *fp;
  fp=fopen("2224.txt","r");
  fscanf(fp,"%d %lf %lf %lf %lf",&n_max, &LL[0], &LL[1], &LL[2], &theta);
  LL[1]=LL[1]/cos(theta);
  //printf("%4d %10.5f %10.5f %10.5f %10.5f\n",n_max, LL[0], LL[1], LL[2], theta);
  for(i=0;i<n_max;i++){
    fscanf(fp,"%lf %lf %lf",&lattice[i][0],&lattice[i][1],&lattice[i][2]);
    //printf("%20.15f %20.15f %20.15f\n",lattice[i][0],lattice[i][1],lattice[i][2]
  }
}

int mk_calc_atom(void){
```

```
  int h,i,i_atom=0,tes=0;
  double max_x=-100.0,min_x=100.0;
  for(h=0;h<n_max;h++){
    if (lattice[h][0]>max_x){
      max_x=lattice[h][0];
    }
    else if (lattice[h][0]<min_x){
      min_x=lattice[h][0];
    }
  }
  for(i=0;i<1000;i++){ calc_atom[i]=-1; }
  for(i=0;i<n_max;i++){
    if ((lattice[i][0]<max_x-x_cut*cos(theta))&&(lattice[i][0]>min_x+x_cut*cos(the
      calc_atom[i_atom++]=i;
      tes+=1;
    }
  }
  //printf("%10.5d\n",tes);
}

double dist(double a[3], double b[3]){
  double dist1=0.0, dd=0.0;
  int i;
  for (i=0;i<3;i++){
    dd=a[i]-b[i];
      //      printf("%10.5f\n",dd);
    dd=dd-round(dd/LL[i])*LL[i];
    //      printf("%10.5f\n",dd);
    dist1 += dd*dd;
  }
    return sqrt(dist1);
}


int reset_NL(void){
  int i,j,i_atom;
  i=0;
  while((i_atom=calc_atom[i++])>=0){
  //  for (i=0;i<n_max;i++){
```

```
    for (j=0;j<20;j++){
      nl[i_atom][j]=-1;
    }
  }
}

int mk_NL(void){
  int i,j,i_nl,i_atom;
  double d1;
  j=0;
  while((i_atom=calc_atom[j++])>=0){
    //  for(i_atom=0;i_atom<n_max;i_atom++){
    //i_atom=0;
    i_nl=0;
    for(i=0;i<n_max;i++){
      d1=dist(lattice[i_atom],lattice[i]);
      if (i!=i_atom){
if ((d1<0.9)){
  nl[i_atom][i_nl]=i;
  //printf("%d %d %10.5f\n",i_nl,i,d1);
  //printf("%10.5f %10.5f %10.5f\n",lattice[i][0],lattice[i][1],lattice[i][2]);
  i_nl++;
}
      }
    }
  }
}

int print_NL(int i_atom){
  int i;
  printf("%3d:",i_atom);
  for (i=0;i<15;i++){
    printf("%3d,",nl[i_atom][i]);
  }
  printf("\n");
}

double lj_energy(int i_atom){
```

```
  double a0,e0,ene,x0,a;
  int i,i_nl;
  a0=1.587401051;
  e0=-1*4.0/12.0;
  ene=0.0;
  for (i_nl=0;nl[i_atom][i_nl]>=0;i_nl++) {
  i=nl[i_atom][i_nl];
    x0=dist(lattice[i_atom],lattice[i]);
    a=1.0/(a0*x0);
    ene+=e0*(pow(a,6.0)-pow(a,12.0));
 }
 return ene;
}

double eam_energy(int i_atom){
  double q,a0,b0,poq,p,rep,rho,x0,E0;
  int i,i_nl;
  //    q=2.00;a0=1.4099023880;b0=2.3747862110;
  //  q=4.00;a0=23.8538969400;b0=9.7680902790;
  //  poq=2.0;p=q*poq;

  //q=2.00;a0=2.899640767;b0=1.781089660;
  //poq=3.0;p=q*poq;

  q=3.00;a0=24.18922289;b0=3.612254621;
  poq=3.0;p=q*poq;

  rep=rho=0.0;
  E0=0.0;
  i_nl=0;
  for (i_nl=0;nl[i_atom][i_nl]>=0;i_nl++) {
    //    printf("%3d\n",nl[i_atom][i_nl]);
    i=nl[i_atom][i_nl];
    x0=dist(lattice[i_atom],lattice[i]);
    //printf("%20.15f\n",x0);
    rep+=exp(-p*x0);
    rho+=exp(-2.0*q*x0);
    E0=a0*rep-b0*sqrt(rho);
    //    printf("%20.15f \n",E0);
```

```
  }
  return E0;
}
double e_total(void){
  int i,j=0,i_atom;
  double e_total=0.0;
  while((i_atom=calc_atom[j++])>=0){
    e_total+=eam_energy(i_atom);
    //printf("%10.5f\n",e_total+=eam_energy(i_atom));
  }
    return e_total;
}

double e_diff(int atom, double dd, int j){
  double e0=0.0,e1=0.0,de;
  int k=0,k2=0,calc_num,l=0;
  e0=eam_energy(atom);

  for(k=0;l>=0;k++){
    l=nl[atom][k];
     for(k2=0;calc_atom[k2]>=0;k2++){
      calc_num=calc_atom[k2];
       if(calc_num==l){
e0+=eam_energy(l);
      }
    }
  }

  //printf("ENE0 %20.15f\n",e0);

  lattice[atom][j]+=dd;
  int k_2=0,k2_2=0,calc_num_2=0,l_2=0;
  //printf("%20.15f\n",lattice[1][j]);
  e1=eam_energy(atom);
  for(k_2=0;l_2>=0;k_2++){
    l_2=nl[atom][k_2];
     for(k2_2=0;calc_atom[k2_2]>=0;k2_2++){
      calc_num_2=calc_atom[k2_2];
       if(calc_num_2==l_2){
```

```
e1+=eam_energy(l_2);
      }
    }
  }

 //printf("ENE1 %20.15f\n",e1);
  de=e0-e1;
  return de;
}

double innerRelax(int mc){
  int h,i=0,j,atom;
  double de,dd;
  srand((unsigned) time(NULL));
  for(h=0;h<mc;h++){
    reset_NL();
    mk_NL();
    atom=0,i=0;
    //printf("------------------------------trial %10.5d\n",h);
    while((atom=calc_atom[i++])>=0){
      for(j=0;j<3;j++){
dd=(1.0-2.0*(rand()%10000/10000.0))/100.0;
de=e_diff(atom,dd,j);
if(exp(de/T)<(rand()%10000/10000.0)){
  //if(de<0.0){
  lattice[atom][j]-=dd;
}
else{
}
      }
    }
  }
}

int outerRelax(void){
  int xx,yy,zz,i;
  int nrange=0;
  int mc=100;
  double e0=0.0;
```

```
  double min_data[4] = {0.0,0.0,0.0,1000.0};
  for(zz=0;zz<=0;zz++){
    for(yy=0;yy<=0;yy++){
      for(xx=1;xx<=1;xx++){
//double dx=(xx/100.0),dy=(yy/100),dz=(zz/100);
double dx=(xx/100.0),dy=(yy/20.0),dz=(zz/20.0);
read_lattice();
mk_calc_atom();
blockmove(dx,dy,dz);
reset_NL();
mk_NL();
innerRelax(mc);
e0=e_total();
printf("%f %f %f %10.5f\n",dx,dy,dz,e0);
if(min_data[3]>e0){
  min_data[0]=dx;min_data[1]=dy;min_data[2]=dz;min_data[3]=e0;
  for(i=0;i<n_max;i++){
    result_lattice[i][0]=lattice[i][0];
    result_lattice[i][1]=lattice[i][1];
    result_lattice[i][2]=lattice[i][2];
  }
}
      }
    }
  }
    printf("result\n%f %f %f %10.5f\n",min_data[0],min_data[1],min_data[2],min_data
}

int blockmove(double dx,double dy,double dz){
  int out;
  for(out=0;out<n_max;out++){
    if(lattice[out][0]>0.0){
      lattice[out][0]=lattice[out][0]+dx;
      lattice[out][1]=lattice[out][1]+dy;
      lattice[out][2]=lattice[out][2]+dz;
    }
  }
}
```

```
int main(void)
{
  clock_t start,end;
  start=clock();
  //read_lattice();
  //mk_calc_atom();
  //reset_NL();
  //printf("Under mk_NL\n");
  //mk_NL();
  //innerRelax();
  outerRelax();
  FILE *fp;
  int i;
  fp=fopen("outlattice.txt","wt");
  for(i=0;i<n_max;i++){
    fprintf(fp,"%20.15f %20.15f %20.15f\n",result_lattice[i][0],result_lattice[i][
  }
  fclose(fp);

  end=clock();
  printf("%.5fsec\n",(double)(end-start)/CLOCKS_PER_SEC);

  return 0;
}
```

付 録C　MapleによるEAMポテンシャルの距離依存性算出コード

```
restart

energy:=z*rep-B*sqrt(z*hop^2);
rep:=A*exp(-p*r);
hop:=exp(-q*r);

energy;
ij_EAM:=unapply(energy,r);

%parameters:
z:=12;
q:=3.0;
poq:=3.0;
p:=q*poq;

ij_EAM(r);

sol1:=solve(diff(ij_EAM(x),x)=0,x);

x0:=sol1[3];

f1:=fsolve({ij_EAM(x0)=-1.0,x0=1.0/sqrt(2)},{A,B});

z:='z';
ij_EAM:=unapply(subs(f1,energy),(r,z));

plot([ij_EAM(x,12)],x=0.2..2.0,y=-1.0..0,labels=["¢()","E(eV)"]);
plot([ij_EAM(x,12),ij_EAM(x,8),ij_EAM(x,4),ij_EAM(x,14)],x=0.2..2.0,y=-1.5..0,labe
```

```
¢()","E(eV)"]);
```