卒業論文
小傾角粒界シミュレーションにおける転位モデルの作成

関西学院大学理工学部
情報科学科 西谷研究室 9553 角田 大知

2013年3月

指導教員　西谷　滋人　教授

概 要

西谷研の八幡は，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為に構築した界面において，静的緩和により安定なエネルギーを原子間ポテンシャルで計算し，原子座標を求めるプログラムを開発した．しかし，そのプログラムからバーガースサーキットや転位を視覚化させることはできなかった．そこで，転位やバーガースサーキットを視覚化できるプログラムの開発をし，シミュレーションを検証することを目的に研究を行った．シミュレーションから得られた結晶の座標ファイルの拡張をはじめ，Ruby プログラムから特定層の原子，バーガースサーキットを作る原子，転位によるズレのある原子，隣接する原子の座標を数値的に抽出し，Processing を用いて視覚化させた．

視覚化させた結果からバーガースサーキットや転位を明らかに視認することが可能になった．傾角 0 °付近の粒界シミュレーション結果は等間隔に垂直なバーガースベクトルを持つ刃状転位が予想通り存在している事が判明した．このことから Read-Shockley の粒界モデルを再現していることがわかった．さらに，研究を進めていく上で傾角９０ °付近の粒界の結晶モデルを視覚化させた際に，緩和させる前の初期座標に不具合があることが判明し，八幡のプログラムに問題があることがわかった．

目 次

第1章　研究背景と目的

小傾角粒界は等間隔に並んだ刃状転位で構成されている．小傾角粒界のモデルである Read-Shockley において傾角 0 °，９０ °付近における Al の粒界エネルギーの角度依存性は，図 1.1 に示すように，立ち上がりの傾きが異なる．しかし，大槻が実験的に観測した角度依存性では傾きが全く同じであるという矛盾が生じている．西谷研の八幡は，この矛盾の原因を解明する為に，構築した界面において，静的緩和により安定なエネルギーを原子間ポテンシャルで計算し，原子座標を求めるプログラムを開発した．しかし，そのプログラムだけでは図 1.2 のように転位の様子を視覚化することはできなかった．シミュレーションが正しければ，小傾角粒界には，等間隔に垂直なバーガースベクトルをもつ刃状転位ができるはずである．つまり，八幡のシミュレーションから転位の様子を明らかにすることで粒子シミュレーションの結果をより詳しく検証することができる．八幡のプログラムでは隣接する結晶モデルと連続した状態であることを考慮した周期的境界条件，x 軸方向の両端の一番外の原子が持つエネルギーの値を安定値の-1 に固定する固定境界条件を実装した上で，外部緩和，内部緩和をすることで粒界を作り，できたモデルの粒界エネルギーを計算している．さらに，そのときにおける原子座標をテキストファイルに出力するように作られている．これをふまえ，本研究では，八幡のシミュレーションにおいて得られる原子座標から転位場所やバーガースベクトルを数値的に取り出し，視覚化させ明らかにすることで，Read-Shockley の粒界モデルに基づいているか否か検証することができると考え，転位を視覚化することができるプログラムを開発し，粒子シミュレーションの結果のより詳しい検証を行う．

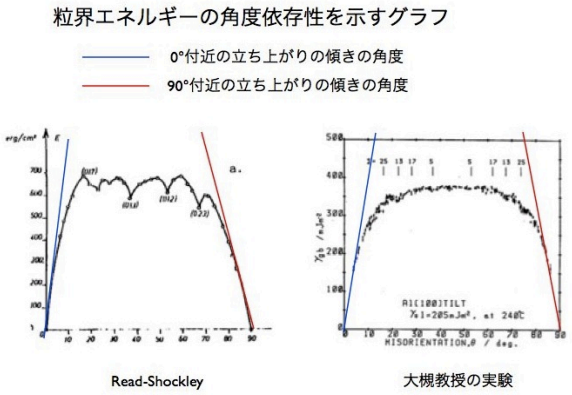

図 1.1: Read_Shockley と大槻教授の実験による角度依存性の違い.

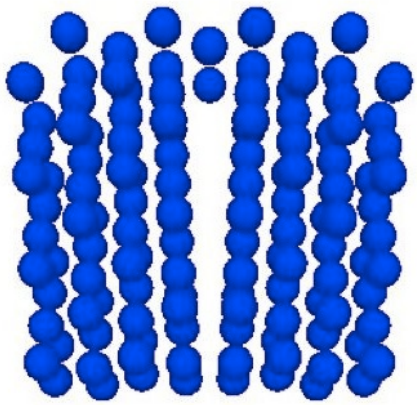

図 1.2: 八幡のシミュレーションから得られる結晶モデル.

1.1　八幡シミュレーションに関する手法

八幡の原子座標を求めるプログラムの中で使用されている手法について説明する．

- 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．そこで周期的境界条件を考慮する必要がある．周期的境界条件とは，図 1.3 に示されるように，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できることである．

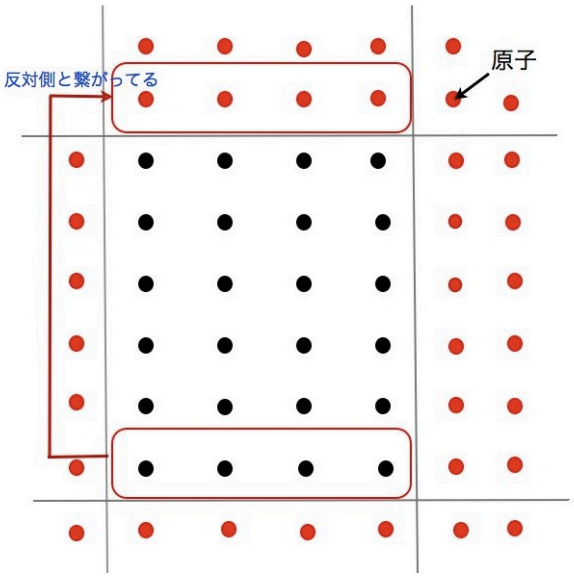

図 1.3: 周期的境界条件.

- 固定境界条件

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 1.4 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，エネルギーの値を安定値の-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，粒界結晶の結晶面から遠い位置にある構造は完全結晶とほとんど同じものであるからである [1].

- 外部緩和

結晶の形や体積を変える外部緩和とは，図 1.5 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 (黒点) の安定位置を模索する手法である．

- 内部緩和

内部緩和とは，図 1.6 のように結晶モデル内の原子 (黒点) を 1 つずつバラバラに動かして再安定位置を求める手法である．

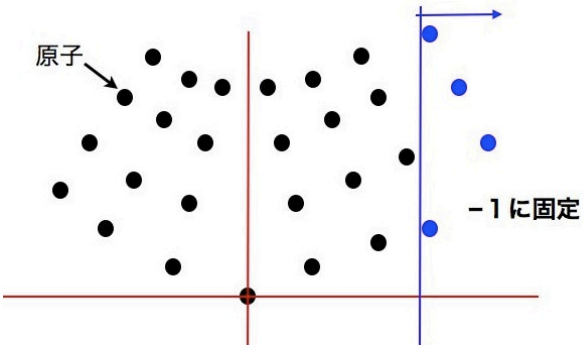

図 1.4: 固定境界条件.

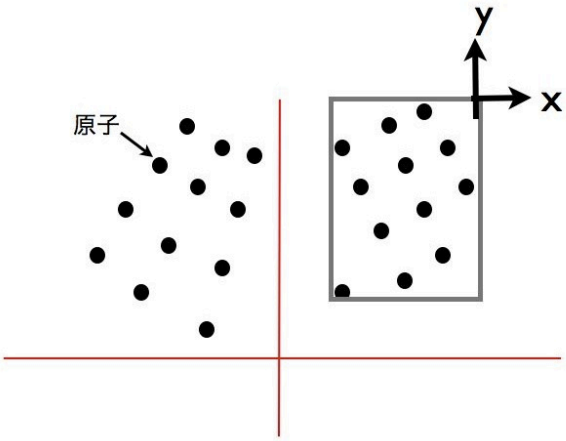

図 1.5: 外部緩和.

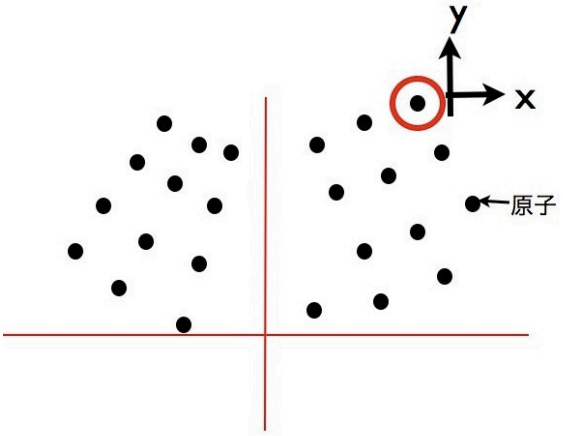

図 1.6: 内部緩和.

第2章　物理学的背景

自然界において，水晶，雪，氷に始まって，エレクトロニクス素子として使われる人工育成されたシリコン単結晶，また固体物理学の研究用資料としての各種単結晶まで様々な結晶が存在する．結晶とは一言で答えるならば，原子あるいは分子が３次元的な周期性を持って規則正しく配列してできた固体のことである [2].

2.1　転位とは

物質は，どれをとっても固体の機械的強さを利用して作られている．ガラスのような特別の物質を除くと固体の大部分は結晶となっている．しかし，結晶の機械的強さで問題となるのは，結晶が急に変化し始めたり，破壊したりする原子またはイオンが集団として動く現象である．すなわち，周期的に並んでいる原子またはイオンの配列そのものを変化させなければならないのである．そこで，変形や破壊の起こる条件の理論の基礎となるものが転位およびその他の格子欠陥である [3].

実際の結晶には何らかの乱れが生じている．それらを総称して格子欠陥と呼ぶ．格子欠陥には０次元的な欠陥の点欠陥や，１次元的な欠陥の線欠陥や２次元的な欠陥の面欠陥がある．線欠陥は転位と呼ばれ，材料の変形を左右する重要な欠陥である．代表的な転位として刃状 (edge) 転位と螺旋 (screw) 転位がある．

刃状転位について述べる．図 2.1 には１原子間距離のすべりがすべり面の左半分で起こっているが，右半分では起こっていないような単純立方格子を示す．すべっている部分とすべっていない部分との境界を転位という．その位置は，図 2.2 に示されるように，結晶の上半分に余分に挿入された垂直な半平面をなす原子面 (extra half plane) の端で示される．転位の近くでは結晶は大きいひずみを受けている．単純な刃状転位は，すべり方向に垂直に，すべり面上をどこまでも続いている [4].

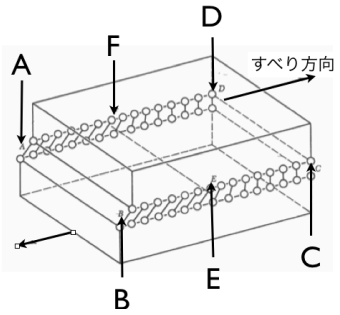

図 2.1: すべり面 ABCD にある刃状転位 EF．原子が格子定数の半分以上変位しているようなすべりの生じている領域 ABEF と，変位が格子定数の半分以下であるすべりの生じていない領域 FECD が示されている．

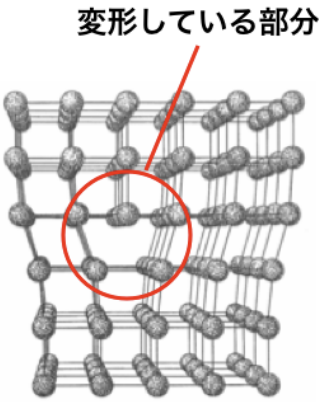

図 2.2: 刃状転位の構造，y 軸の上半分に余分の原子価を挿入した為に，変形していると考えられる．この挿入で上半分の原子は圧縮され，下半分の原子は引き伸ばされる．

2.2　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 2.3 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，それらの隣接している部分には境界ができ，それを粒界という．

対称傾角粒界とは図 2.4 のように二つの立方晶が回転角 θ 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 °程度を小傾角粒界という．

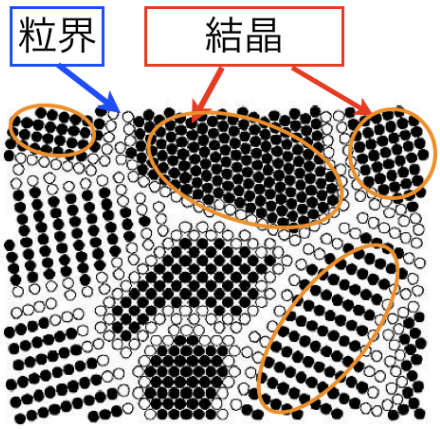

図 2.3: 多結晶における粒界.

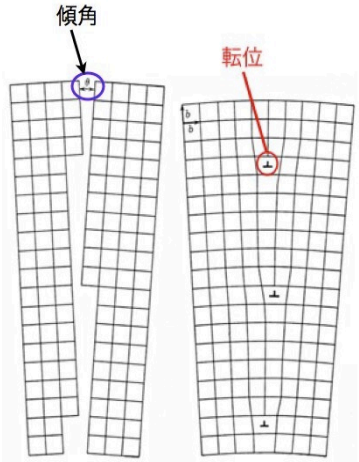

図 2.4: [100] 対称傾角粒界.

第3章　手法

3.1　プログラミング使用言語について

今回の研究において２つのプログラミング言語を使っていて，その使い分けを以下に示す.

Processing

Processing は，アートとデザインのためのプログミングを実現オープンソース開発環境として Ben Fly 氏 Casey Reas 氏によって開発されました．Windows にも Mac にも，さらに Linux にも対応していて，無料でダウンロードして利用することができる．シンプルで軽く，はじめての人にもとても使いやすい環境である [5]. よって粒界のモデルを視覚化するようにしているのは Processing で行っている．

Ruby

Ruby はまつもとゆきひろ氏によって開発されたオブジェクト指向のスクリプト言語の名称である．全てのデータは一貫してオブジェクトとして表現されているので考えたことを素直に記述できるようになっていて，継承や Mix-in といった，オブジェクト指向言語らしい機能はもちろん入っている．また，様々なクラスライブラリが標準で添付されているほか，例外によるエラー処理や，自動的にメモリの解放を行うガベージコレクタなど，快適なプログラミングを支援する機能も備わっている [6].

今回の研究において八幡の開発したプログラムによって得られた座標のテキストファイルを読み込み，モデル作成に必要なファイルの作成をこの Ruby 言語により作成するようにしている.

3.2　バーガース･ベクトル (Burgers vector) とバーガース・サーキット (Bugers circuit)

実際の結晶は熱振動，不純物原子，空格子点などの他，弾性歪の為に理想的な原子の配列からずれている．これらの乱れを除いた結晶を理想結晶とする．実際の結晶と理想結晶とを比較すると両方の結晶中のひとつひとつの原子が，ある領域では１対１に対応させられるだろうが，他のある領域では，実際の結晶の原子の位置が非常に乱れて，理想結晶のどの原子の位置に対応させてよいかわからな

い領域もある．実際の結晶のある領域中の原子を，理想結晶の原子の位置と１対１に対応させることができるならば，結晶はその領域内で完全結晶であるといい，対応できない結晶の部分を不完全結晶という．

実際の完全結晶内の原子を，次々と結んで，閉じた１つの経路を作る．これをバーガース・サーキットと呼ぶことにする．理想結晶にも対応する経路を考えることができる．この理想結晶中の対応する経路を対応サーキットと名付ける．

今，図 3.1 のように，完全結晶におけるバーガース・サーキットがあるとすると，対応サーキットもまた閉じている．しかし，もしバーガース・サーキットが不完全結晶を含んでいる面を取り巻く場合には，バーガース・サーキットの近くは完全結晶であっても，対応サーキットは閉じるとは限らない．そこで対応サーキットが閉じないとき，バーガース・サーキットは転位を囲むと考えられる．バーガース・サーキットを時計の針の進む方向と逆にとったとき，対応サーキットを閉じさせるのに必要なベクトル (対応サーキットの終点から始点へのベクトル) をバーガース・ベクトルと呼ぶ．正の刃状転位の周りのバーガース・サーキット，対応サーキットをそれぞれ図 3.2 に示す．[3]

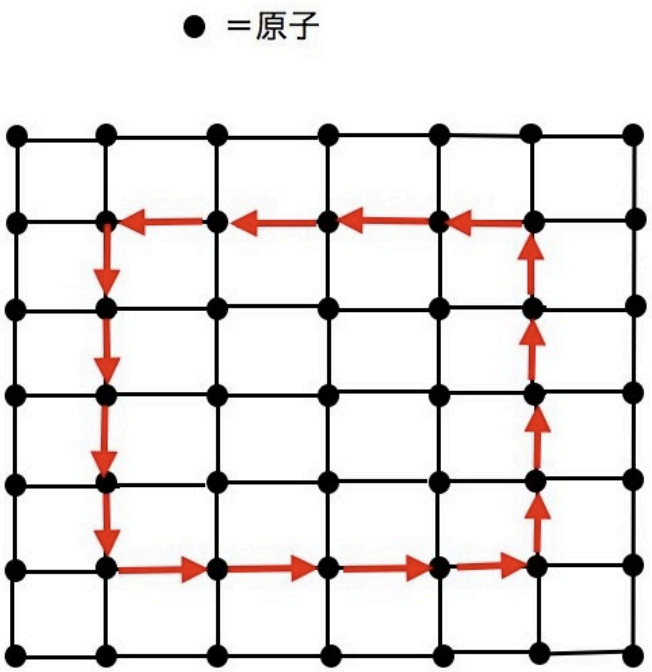

図 3.1: 完全結晶内のバーガースサーキット．

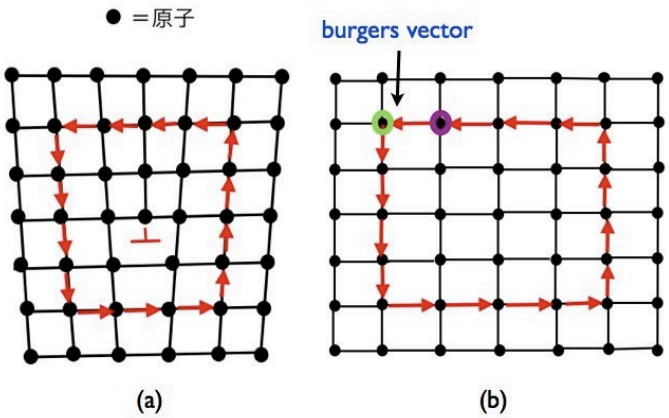

図 3.2: 刃状転位まわりのバーガース・サーキット〔a〕と対応サーキット〔b〕.

第4章　視覚化用複数ファイル作成

4.1　シミュレーションまでの流れ

八幡のプログラムの結果から得られたユニットセルの原子座標ファイルからシミュレーションまでの流れは (図 4.1) のようになる．八幡のシミュレーションから得られる原子座標はあくまでもユニットセルであり等間隔に刃状転位を持つ粒界モデルを作成するには結晶が連続した状態を仮定するため，原子座標を拡張する必要がある．そこで原子座標を拡張するプログラムの開発を行い，拡張させた原子の中でバーガースサーキットを作る原子などの情報を tsv のファイルに格納する．最後に Processing にて tsv のファイルを読み込み視覚化させる．

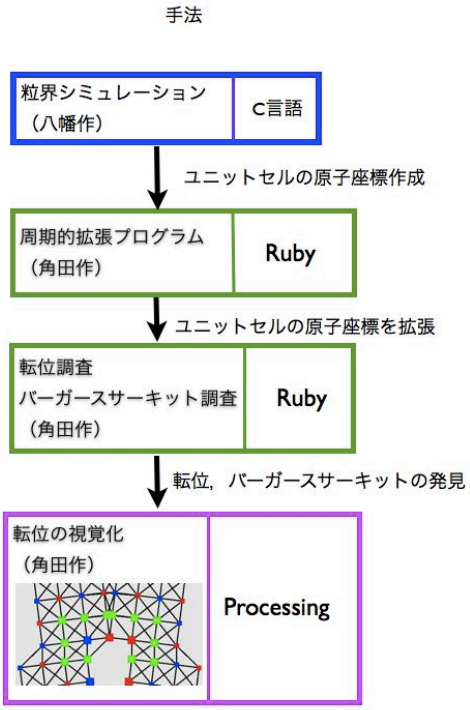

図 4.1: シミュレーションまでの流れ.

4.2　周期的拡張プログラムの説明

八幡が作成したシミュレーションから得られる原子座標はひとつの結晶における原子数分しかない．しかし実際，物質は結晶が前後上下左右に連なった形で生成されている．つまり，八幡のプログラムから得られた座標ファイルのみでシミュレーションを行うと前後，上下，左右の結晶とのつながりがわからない．よって，結晶は周期的境界条件から成り立っているので，元の結晶の座標ファイルを拡張して，前後，上下にも同じ結晶を作ることで結晶同士の端の原子の結びつきが視覚化できるようにする (図 4.2)．今回，八幡のプログラムでは x 軸方向に関して固定境界条件を適用しているため，左右の結晶のつながりに関しては無視する．手法として結晶における前後，上下における最大値と最小値を見つけ，その差と隣接する原子間の距離の和を全ての原子に足す．もしくは引くことで拡張することができる．プログラムの流れは図 4.3 のようになる．

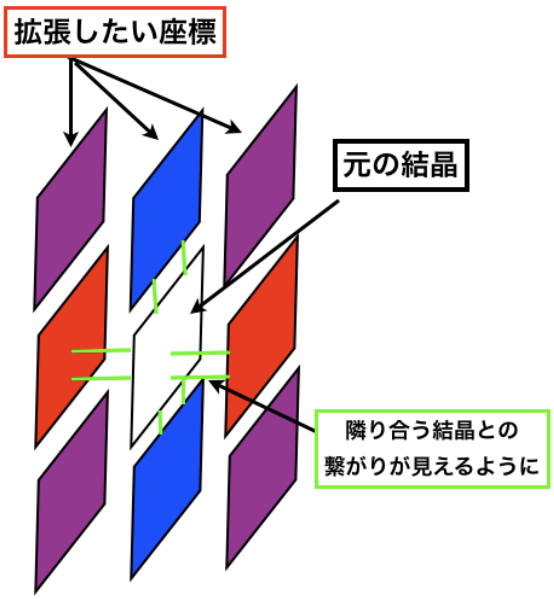

図 4.2: 拡張させたい座標.

初めに data=File.read(”元の座標ファイル名”) で八幡のプログラムによって得られた結晶の座標ファイルを読み込む．その座標ファイルの一行目には，目的とするファイル作成において不必要な情報 (図 4.4) が入っている為に 1 行目は読み込まないようにしている．

次に座標の最大値と最小値を求める関数を作り，元のユニットセルにおける．y 方向，z 方向における最大値と最小値を求める．その差と原子間の距離を足し合わせた値 (図 4.5 のイメージ) を各原子に加減算することで結晶を拡張したファイルを作成する．元の結晶に対して前後，上下に拡張するため，計 9 つ分の結晶ができる．

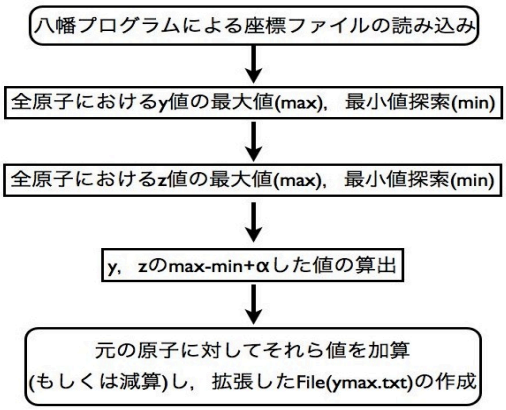

図 4.3: プログラムの流れ.

142 100.000000 5.000000 2.000000 7.125016
-1.486486 -0.217061 0.000000
-1.548504 0.279078 0.500000
-0.990347 -0.155043 0.500000
-1.052365 0.341096 0.000000
-1.486486 -0.217061 1.000000
-1.548504 0.279078 1.500000
-0.990347 -0.155043 1.500000
-1.052365 0.341096 1.000000
-1.610521 0.775217 0.000000
-1.672538 1.271356 0.500000
-1.114382 0.837234 0.500000
.
.
.

図 4.4: 八幡プログラムによって得られる座標ファイルの例.

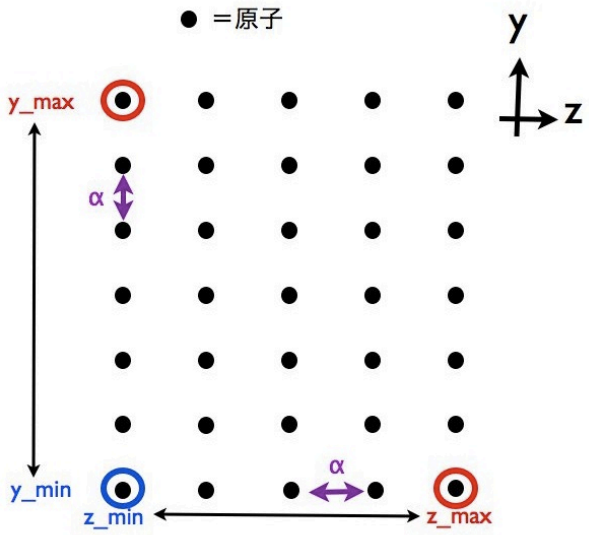

図 4.5: 値の取り出すイメージ.

4.3 転位調査プログラムの説明

周期的拡張プログラムである edit_txt.rb によって編集された座標ファイル ymax.txt を読み込み，座標を$atom という配列変数に入れている．初めに All_atom.tsv のファイルに，z 軸に対して指定させた層内での全原子の座標を入れている．後述する関数などを用いてシミュレーションに必要なファイルの作成を行っていく．

4.4 Ruby プログラムにおける関数の説明

4.4.1 tsv ファイル用ダミー挿入関数

Processing に引き渡すファイルに初めの一行は読み込まれないのでダミーとして 0 の値を入れる関数を作った．整数の引数 n を受け取りファイルの一行目に n 列分の 0 をダミーとして挿入する．以降ファイルの作成に関しては必ずこの関数を用いてダミーを初めに挿入させている．

4.4.2 原子間距離を測る

原子間距離を測る為の関数 distance を作成する．図 4.6 のように引数として受け取った 2 点間の距離をこの関数により調べ，配位数の判定，隣接原子の座標ファイル作成，バーガースサーキットを作る原子を取り出す際に，この関数を大いに活用する．

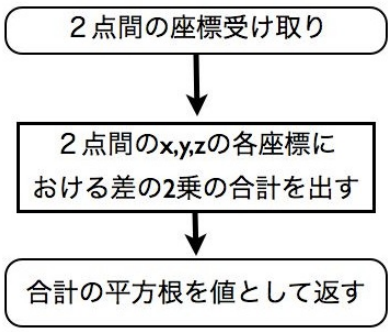

図 4.6: distance 関数の流れ.

4.4.3 原子間のボンドを引く為のファイル作成

以下の distance_bond() 関数は原子間のボンドを引く為の元となるファイル作成の為の関数である．全ての原子に対して原子間距離を測定し，指定した距離以内かつ z 軸に対して指定した層内でのボンドを引くようにしている．２点間の x,y,z 座標と初めの１列目はダミーで値０を入れるので計７列のファイルになっている．８列目に２点間の距離を表示するようにしているが８列目はファイルを Processing に渡す際に影響しない（図 4.7）．この関数の中で原子間距離が最も小さい値を探してきている，その値を$shortest という変数に入れている．この値は後述するがバーガースサーキットを探すコードの中で必要になってくる．この流れを模式的に示すと以下のようになる（図 4.8）.

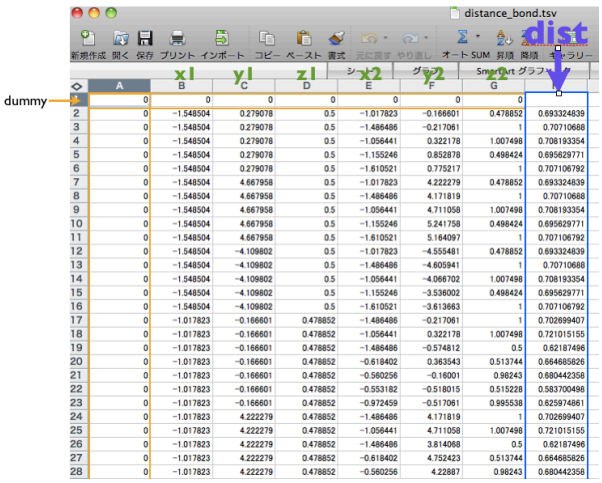

図 4.7: ファイルの例.

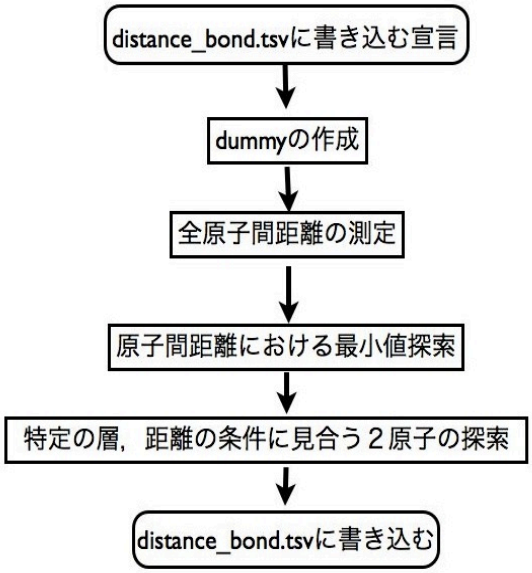

図 4.8: コードの流れ.

4.4.4 配位数が 11 以下の原子を見つける

次の find_dislocation 関数では転位によるズレがある為に配位数の数が不完全な原子取り出している．原子の数だけ配列を用意し，全ての原子間距離を測定した上で距離の条件に見合う原子数を要素として配列に入れる．完全結晶ならば面心立法 (fcc) 構造の配位数が 12 なので，ズレが生じている原子の配位数は 11 以下になっているはずである．つまり，要素数が 11 以下の原子は転位によるズレがあると考えられるので，その原子番号を$dislocation の変数に入れている．上記のコードの流れは以下のように模式的に示される（図 4.9）.

source code

```
def find_dislocation()
  bond=[]
  for i in 0..$atom.size-1 do
    bond[i]=[]
  end
  for i in 0..$atom.size-1 do
    atomi=$atom[i]
    for j in 0..$atom.size-1 do
      atomj=$atom[j]
      dist = distance(atomi,atomj)
      if (dist>=$dist_min and dist<$dist_max) then
        bond[i] << j
      end
    end
    if bond[i].size<=11 then
      $dislocation << i
    end
  end
end
```

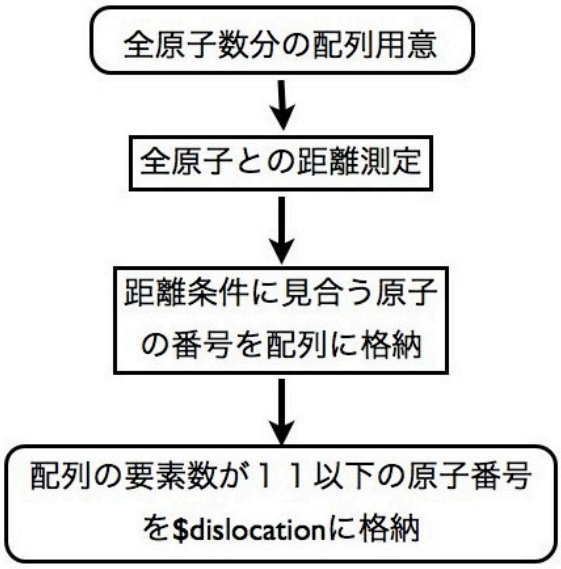

図 4.9: コードの流れ.

4.4.5 特定範囲内の原子を抽出する関数

specify_area 関数では特定範囲内での原子を抽出するために使用する関数である．この関数により，変数$dislocation に入っている，転位によるズレがあると考えられる原子をさらに特定の範囲の中から絞ることができる．特に今回のモデルでは x 座標における端の原子は固定境界条件より更なる結晶への繋がりを考えていないため必然的に配位数が少ないと考えられる．そこでこの関数より端の原子を除くようにしている (図 4.10 のイメージ)．特定範囲内に絞られた原子の番号を a という配列の変数に入れて返している．

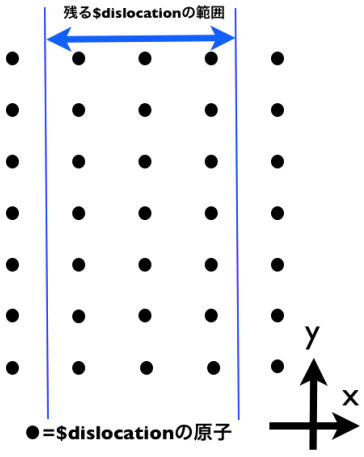

図 4.10: x の範囲を特定させたときのイメージ.

4.4.6　特定範囲内の配位数 11 以下の原子のファイル作成

次の red_point_dislocation() の関数では配位数が 11 以下の原子が見つけられた後，specify_area 関数でその原子を特定範囲内で絞り，Processing で表示させる為の原子座標ファイル作成している（図 4.11）．また$dislocation2 という配列を用意し，$dislocation の中に入っているリストを入れる．この$dislocation2 の配列は後述するバーガースサーキットを視覚化させる為に必要になる．

source code

```
def red_point_dislocation()
  data=[]
  $dislocation2=[]
  file=open("3D_dislocation.tsv" ,"w")
  mk_dammy(file,4)
  for i in 0..$dislocation.size-1 do
    if $atom[$dislocation[i]][2]>$z_specify1 && $atom[$dislocation[i]][2]<$z_specify2
      file.print  0.to_s+"\t"+$atom[$dislocation[i]][0].to_s+"\t"
+$atom[$dislocation[i]][1].to_s+"\t"+$atom[$dislocation[i]][2].to_s+"\n"  ##(dammy,x,y,z)
      data.push([0.to_f,$atom[$dislocation[i]][0].to_f,
$atom[$dislocation[i]][1].to_f,$atom[$dislocation[i]][2].to_f])
      $dislocation2<<$dislocation[i]
    end
  end
  pp $dislocation2.size
end
```

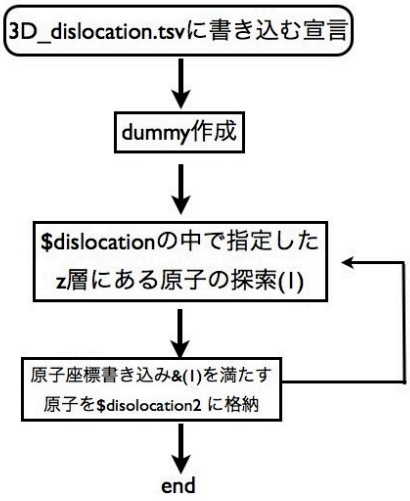

図 4.11: ファイルの内情報.

4.4.7　バーガースサーキットを作る原子リストの作成

次に mk_burgers の関数においてバーガースサーキットを視覚化させる為に，転位を囲む原子の座標ファイルを作成することを目的とする．まず，転位によるズレがあると思われる原子のリスト$dislocation2 にある原子の y 座標から最小原子間距離である$shortest の$rate 倍の前後幅において，$dislocation2 の原子が他にあるか探す．図 4.12 に示されるように同層に 3 つ以上の原子がある部分に明らかに転位があると考えられるため，同層にあるだろう原子のリストを teni という配列の中に格納する．

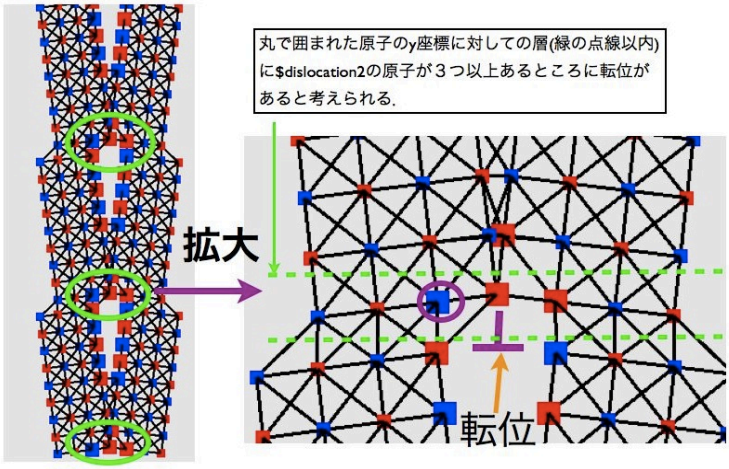

図 4.12: バーガースサーキットの原子を取り出す step1.

次に，先ほど teni の配列に格納された原子を同層毎のグループにさらに分ける作業をする．上記と同様に，teni の中にある原子で同層にあると思われる原子でグループを作る．この時，同層にある原子を配列 group に入れる度，teni から group に入った原子番号は削除し，teni の要素が 0 になるまで繰り返す．

source code

```
  count=0
  while teni.size!=0 do
    group[count]=[]
    for j in 0..teni.size-1 do
      if ($atom[teni[count]][1]+($shortest*$rate) > $atom[teni[j]][1]
&& $atom[teni[count]][1]-($shortest*$rate) < $atom[teni[j]][1])
        group[count] << teni[j]
      end
    end
    for z in 0..group[count].size-1 do
      teni.delete(group[count][z]) # this code delete same number of group[count]
     from teni's array
    end
    count=count+1
  end
  pp group
```

次に group の配列に入っている原子群が転位の部分にあたるのでその周りの原子をとりだす作業を行う．各 group の原子周りにある第一近接原子を配列 nl_group にいれる (図 4.13)．この時，$dist_max 以内の原子を nl_group の配列に入れるのだが，自らの原子も入ってしまうので，同じ番号の原子を nl_atom の中から削除する．以上で第一近接原子のみが取り出されたのでそれらの原子を mk_burger.tsv に書き込むことでファイルを作成する．この流れを模式的に示したのが図 4.14 のようになる．以上の関数の中で行っている流れを (図 4.15）に示した．

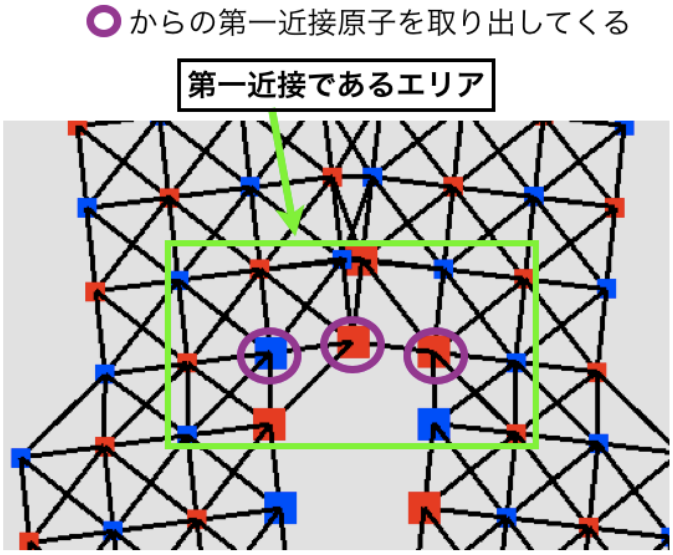

図 4.13: バーガースサーキットの原子を取り出す step2.

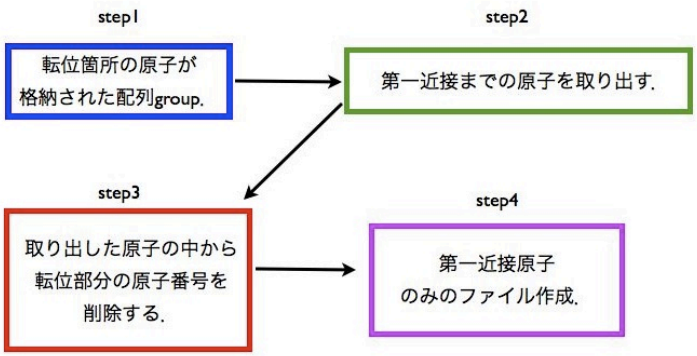

図 4.14: 第一近接原子を取り出すまでの step.

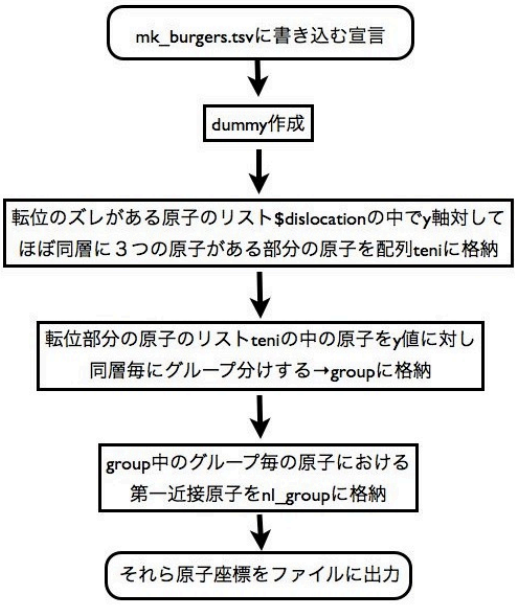

図 4.15: mk_burgers() 関数の流れ.

4.5 転位調査プログラム実行の過程

4.5.1 実行の流れ

main となる実行文では，ユニットセルを拡張した座標ファイルを読み込みユーザーにボンドを引く為の原子間距離の条件$dist_max と特定層を取り出す為の条件値$z_specify1 を入力させる．$dist_max をユーザーからの入力に任せるのは，緩和したモデルによって原子間の距離が変わるので一定の値でモデルを作ると原子間が引かれてないモデルが作成されてしまう可能性がある．つまり，適当な値を入力することによって各原子間にボンドの引かれたモデルの作成をする．$z_specify をキー入力させる理由として．緩和前の結晶モデルでは原子の z 値が 0 から 1.5 までに 0.5 区切りに 4 層存在していた．その中の 0.5，1.0 付近の 2 層を取り出すことによってプログラムが通り実行されるようになっていた．しかしここで，緩和すると図 4.16 のように原子が移動し層が内に寄ることがある．つまり，中の 2 層が取り出せるようにユーザーに適当な値を$z_specify1 に入力させている．その値を 1.5 から引くことで特定層を取り出す為のもう一方の値$z_specify 2 を定義している．取り出す層内での全ての原子座標を All_atom.tsv に書き込む．その後，以上の関数を使って，隣接原子間のボンドを引く為のファイル，配位数 11 以下の原子のファイル，バーガースサーキットを作る為の原子のファイル作成をする．この流れを模式的に表したのが図 4.17 である．

プログラムを実行させたときの実行文とターミナル上での出力は図 4.18 のようになる．

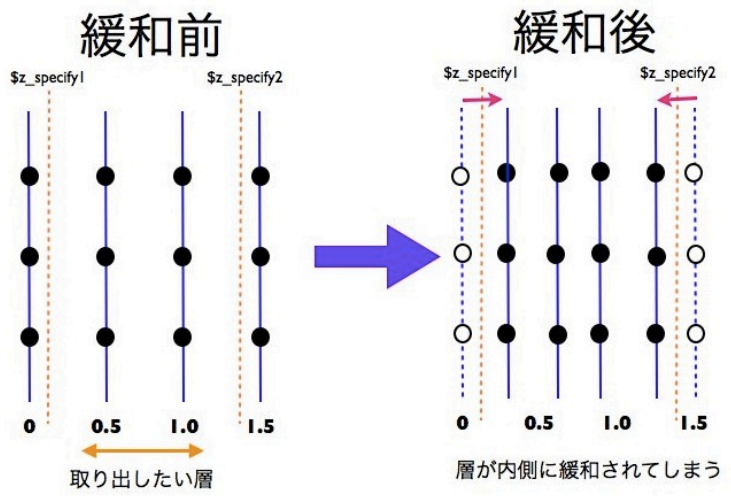

図 4.16: 層の緩和による変化.

図 4.17: 実行プログラムの流れ.

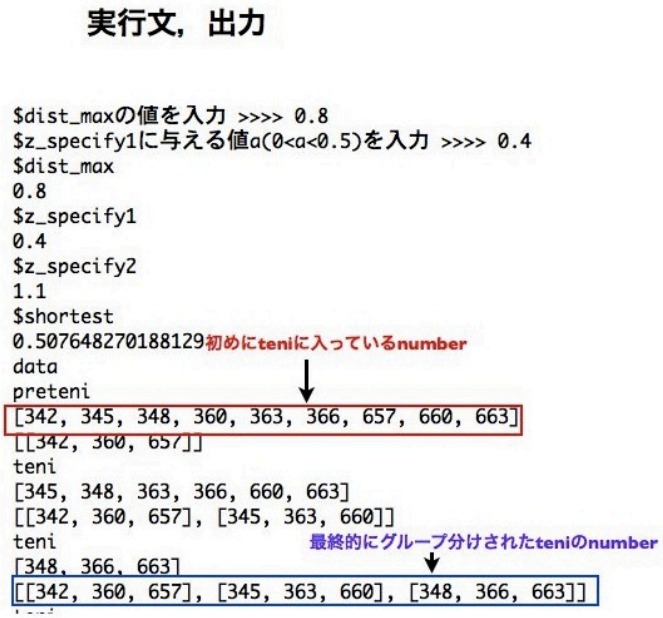

図 4.18: ターミナル上での入力と出力.

第5章　視覚化システム

Ruby のプログラムで作成したいくつかの tsv ファイルを次の Processing でつくったプログラムに読み込ませることでモデルとして表示させることができる．Processing で作ったコードには立体的に原子の奥行きがわかる３次元のものと全ての原子を平面上に示した２次元のものの２種類を作った．この２つの主なプログラムの構成は同じであるが描画する際に z 座標のパラメータをリアルな数値か 0 にするかという分け方をした．

5.1　Processingにおける視覚化までの流れ

1. まず, 配列変数として定義した原子座標ファイルを変数 data に入れる．実際の数値を表示させようとしても値が小さすぎる為に全原子が密集して見えてしまう．そこでファイル内の各値に size を倍してやることで表示させた際に密集することがなくモデルとして見やすくなる．以上を，特定層内における原子座標のファイル，原子間にボンドを引く為に作成されたファイル，配位数が 11 以下の原子座標のファイル，バーガースサーキットを作る原子座標のファイルに同様の作業を行い，ファイルの読み込みを完了させる．

2. 次に，Processing で線や点を表示させる際に元々メソッドとして備わっている line() と point() を使う．線を引く為に２点間座標を line(x1,y1,z1,x2,y2,z2) として用いる．原子自体は point() を使い，point(x1,y1,z1) で描画する．stroke(0,0,0) は色を表し，中の数字の変化で色が変わる．strokeWeight() は線の太さを示している．

3. 表示させるモデルにおいて，横から見れば当然どの原子が手前にあるのかがわかるが正面から見た際，どの原子がどの原子とどういう繋がりをしているかがわからない．そこで，どの原子が奥にあるのか手前にあるのかをよりわかりやすくする為に原子に色分けをする．表示させる原子中の z 座標における最大値と最小値の差を取り出し，各原子の z 座標から最小値を引き，差で割った値に色の最大値 255 をかけ青を示すようになっている．赤色にするには先ほどの分子を１から引く事で手前が赤色，奥に行くほど青色になるようになっている．この仕組みは図 5.1 のようになる．以上までを視覚化した際の層を横から見た図 5.2 と正面からの図 5.3．プログラムのコードは以下のようになる．

source code

```
z_max=data_atom[0][2];
z_min=data_atom[0][2];
for (int row = 0; row < atom-1; row++) {//tsv_gyou
    if (z_max<data_atom[row][2]){
z_max=data_atom[row][2];}
    if (z_min>data_atom[row][2]){
z_min=data_atom[row][2];}
}
l=z_max-z_min;
for (int row = 0; row < atom-1; row++) {//tsv_gyou
    nz1 = data_atom[row][2];
    R1=int(255*(1-(nz1-z_min)/l));
    B1=int(255*(nz1-z_min)/l);
    strokeWeight(12);
    stroke(R1,0,B1);
    point(data_atom[row][0]*zoom, -data_atom[row][1]*zoom, -data_atom[row][2]*zoom);
}
```

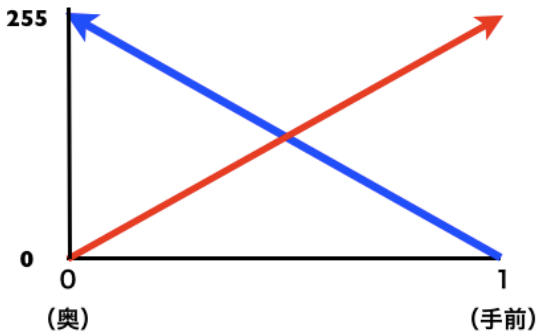

図 5.1: 色の変化を模式的に表したもの.

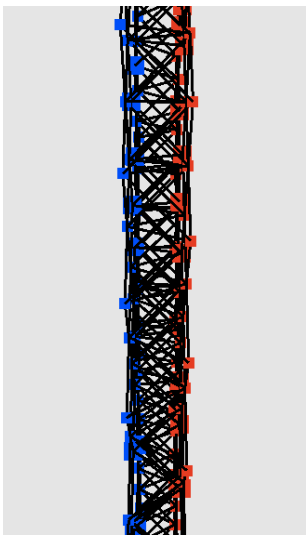

図 5.2: 3 次元で取り出した層の横から見たモデル.

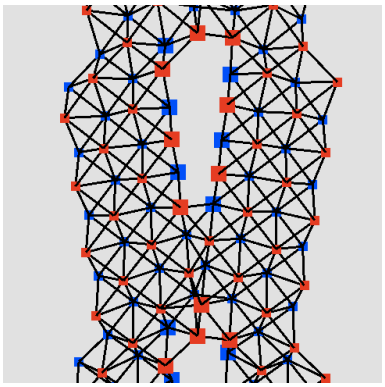

図 5.3: ３次元のモデルを正面から見たモデル.

最後にバーガースサーキットとなる原子を緑色で描画している (図 5.4). プログラムの中でシミュレーションしたモデルを拡大し原子同士の繋がりをよりわかりやすく視覚化できるように，モデルの回転，拡大するコードを加え改良している.

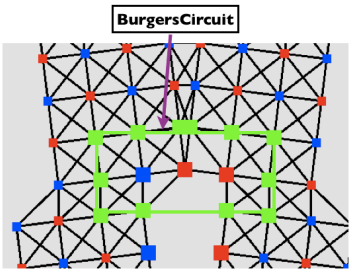

図 5.4: バーガースサーキットを平面的に表したもの.

第6章　議論

八幡シミュレーションから得られた傾角０°付近の原子座標ファイルを２つ使ってモデルを作成した．ファイルのひとつは緩和していないもの (dummy.txt)，もう一方は緩和したものである (Dummymc100.txt)．拡張したモデルの全体像は図 6.1 のようになる．ユニットセルを周期的に拡張したため転位が等間隔に並んでいる．この２つのモデルの転位部分をアップした画像が次の図 6.2 であり，平面図で表したのが図 6.3 になる．

これらの図から，転位部分とバーガースサーキット，バーガースベクトルが容易に視認できた．つまり，八幡のシミュレーション結果は，等間隔に垂直なバーガースベクトルを持つ刃状転位とそれを囲むバーガースサーキットが予測通り存在している事が判明した．よって Read-Shockley の粒界モデルを再現していることがわかった．以上より，傾角０°付近における八幡のシミュレーションに問題はないと考えられる．

一方，傾角９０°付近における八幡のシミュレーションは図 6.4 のようになった．この図は明らかに，Read-Shockley の粒界モデルに基づいているようには見受けられない．そのことから緩和させる前の初期座標が幾何学的におかしいことが判明し，八幡のプログラムに問題があることがわかった．

このプログラムは八幡のシミュレーションによって得られた原子座標から視覚化させることを目的に作成されている．今後は，他の転位モデルにおいても視覚化させることができるように汎用性を追求する必要がある．また，原子間距離の$dist_max，特定層を取り出す$z_specify1 の自動算出を可能にし操作性を高めることが必要であると考えられる．

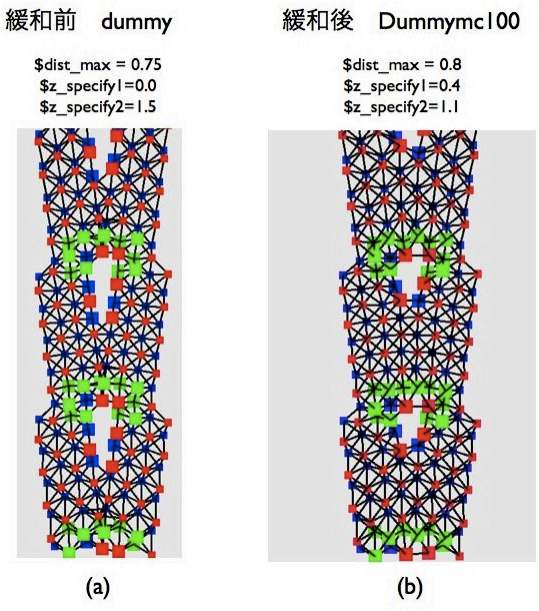

図 6.1: 緩和前，緩和後のモデルの全体像.

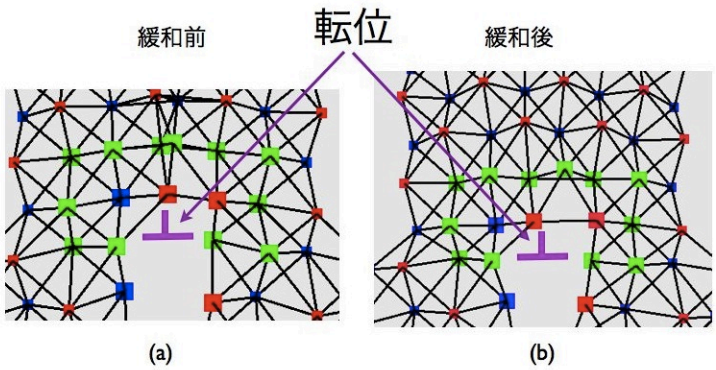

図 6.2: 二つのモデルにおける転位部分の立体図.

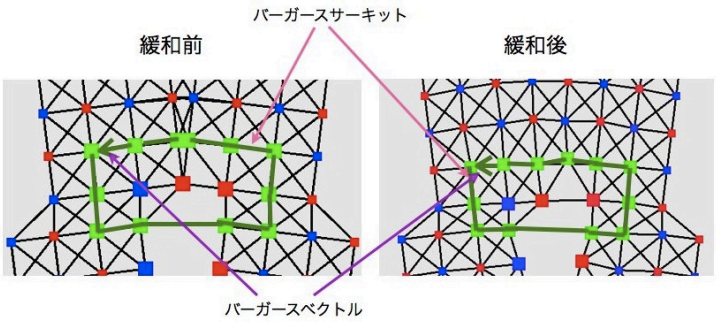

図 6.3: 二つのモデルにおける転位部分の平面図.

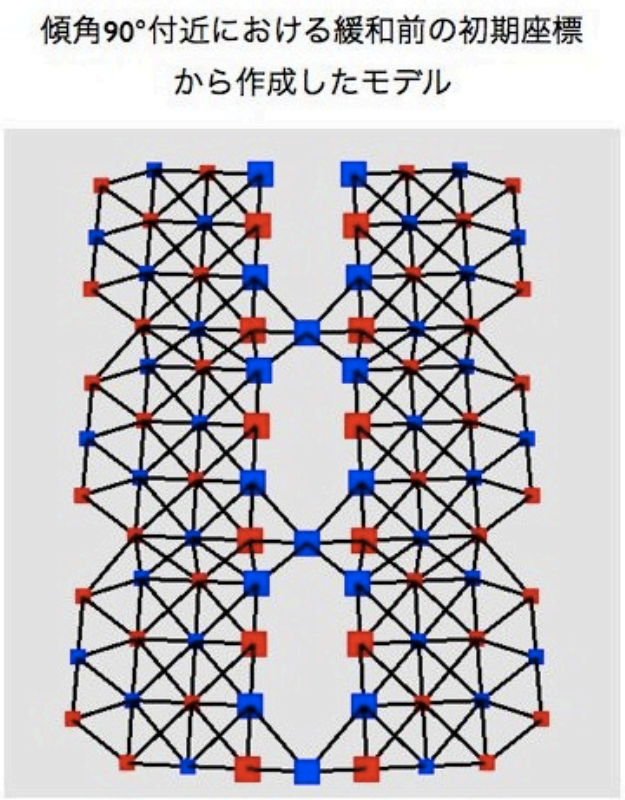

図 6.4: 傾角 90 °付近の粒界モデル.

第7章　総括

本研究では，八幡のシミュレーションから得られた原子座標を用いて転位やバーガースサーキットを視覚化できるプログラムの開発をし，シミュレーションを検証することを目的に研究を行った．シミュレーションから得られた結晶の座標ファイルの拡張をはじめ，Ruby プログラムから特定層の原子，バーガースサーキットを作る原子，転位によるズレのある原子，隣接する原子の座標を数値的に抽出し，Processing を用いて視覚化させた．

視覚化させた結果からバーガースサーキットや転位が明らかになり，傾角０°付近の粒界シミュレーション結果は等間隔に垂直なバーガースベクトルを持つ刃状転位が予想通り存在している事が判明した．このことから Read-Shockley の粒界モデルを再現していることがわかった．一方，研究を進めていく上で傾角９０°付近の粒界の結晶モデルを視覚化させた際に Read-Shockley の粒界モデルに基づいていなかった．そのことから，緩和させる前の初期座標に不具合があることが判明し，八幡のプログラムに問題があることがわかった．

今後，他の転位モデルにおいても視覚化させることができるように汎用性を追求する必要があり，原子間距離の$dist_max，特定層を取り出す$z_specify1 の自動算出を可能にし操作性を高めることが必要であると考えられる．

関連図書

[1] 八幡裕也,「小傾角粒界シミュレーション」,(関西学院大学情報科学科卒業論文 2012, 13)

[2] 黒田登志雄,「結晶は生きている」,(サイエンス社 2006，8)

[3] 鈴木秀次,「転位論入門」,(アグネ　1967,1-3).

[4] 釜下翔,「転位の原子レベルでの視覚化」,(関西学院大学情報科学科卒業論文 2011，3)

[5] 三井和男,「デザイン言語 Processing 入門」,(森北出版株式会社　2011，i).

[6] 高橋征義，後藤裕蔵,「たのしい Ruby」,(ソフトバンククリエイティブ株式会社 2006, viii-ix)

第8章　謝辞

本研究の遂行にあたり，終始多忙な有益な御指導，及び丁寧な助言を頂いた西谷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先輩方，並びに同輩達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．この場を借りて深く御礼申し上げます．

付 録A

A.1 周期的拡張プログラム

source code

```
include Math
require 'pp'
datacheck=true
data=File.read("dammymc100.txt")
$atom=[]
data.each do | line |
  if datacheck then datacheck=false; next end
  line1=line.split(" ") #devide_space
  $atom << [ line1[0].to_f , line1[1].to_f, line1[2].to_f]
end
puts data
def yz_max(n)
  max=$atom[0][n]
  for i in 0..$atom.length-1 do
    if max<$atom[i][n] then
      max=$atom[i][n]
    end
  end
  return max
end
def yz_min(n)
  min=$atom[0][n]
  for i in 0..$atom.length-1 do
    if min>$atom[i][n] then
      min=$atom[i][n]
    end
  end
  return min
end
ymax=yz_max(1)
ymin=yz_min(1)
zmax=yz_max(2)
zmin=yz_min(2)
b=0.3
c=0.3
y_gap=ymax-ymin+b
z_gap=zmax-zmin+c
file=open("ymax.txt" ,"w")
for i in 0..$atom.size-1 do
  file.print  $atom[i][0].to_s+"\t"+$atom[i][1].to_s+"\t"+$atom[i][2].to_s+"\n"
  file.print  $atom[i][0].to_s+"\t"+$atom[i][1].to_s+"\t"+($atom[i][2]+z_gap).to_s+"\n"
  file.print  $atom[i][0].to_s+"\t"+$atom[i][1].to_s+"\t"+($atom[i][2]-z_gap).to_s+"\n"
  file.print  $atom[i][0].to_s+"\t"+($atom[i][1]+y_gap).to_s+"\t"+$atom[i][2].to_s+"\n"
  file.print  $atom[i][0].to_s+"\t"+($atom[i][1]+y_gap).to_s+"\t"+($atom[i][2]+z_gap).to_s+"\n"
  file.print  $atom[i][0].to_s+"\t"+($atom[i][1]+y_gap).to_s+"\t"+($atom[i][2]-z_gap).to_s+"\n"
  file.print  $atom[i][0].to_s+"\t"+($atom[i][1]-y_gap).to_s+"\t"+$atom[i][2].to_s+"\n"
  file.print  $atom[i][0].to_s+"\t"+($atom[i][1]-y_gap).to_s+"\t"+($atom[i][2]+z_gap).to_s+"\n"
  file.print  $atom[i][0].to_s+"\t"+($atom[i][1]-y_gap).to_s+"\t"+($atom[i][2]-z_gap).to_s+"\n"
end
```

A.2　転位発見プログラム

source code

```
include Math
require 'pp'
require 'scanf'

def mk_dammy(file,n)
  a=0.to_s
  for i in 0..n-2 do
    a+="\t"+0.to_s
  end
  file.print a+"\n"
end

def specify_area(b,c,d) #b=min_ratio,c=max_ratio,d=x(0) or y(1) or z(2)
  a=[]
  max_range = $atom[$dislocation[0]][d]
  min_range = $atom[$dislocation[0]][d]
  for j in 0..$dislocation.size-1 do
    if max_range<$atom[$dislocation[j]][d]
      max_range=$atom[$dislocation[j]][d]
    end
    if min_range>$atom[$dislocation[j]][d]
      min_range=$atom[$dislocation[j]][d]
    end
  end
  for i in 0..$dislocation.size-1 do
    if $atom[$dislocation[i]][d]>=min_range*b && $atom[$dislocation[i]][d]<=max_range*c
      a << $dislocation[i]
    end
  end
  return a
end

def distance(atomi,atomj)
  tmp=0
  for i in 0..2 do
    x=(atomi[i]-atomj[i])
    tmp+=x*x
  end
  return sqrt(tmp)
end

def distance_bond()
  file=open("distance_bond.tsv" ,"w")
  mk_dammy(file,7)
  $shortest=10
  for i in 0..$atom.size-1 do
    atomi=$atom[i]
    for j in i+1..$atom.size-1 do
      atomj=$atom[j]
      dist = distance(atomi,atomj)
      if dist < $shortest then $shortest=dist end
      if (dist<$dist_max) && (atomi[2]>$z_specify1 && atomi[2]<$z_specify2)
 && (atomj[2]>$z_specify1 && atomj[2]<$z_specify2) then
        file.print  0.to_s+"\t"+atomi[0].to_s+"\t"+atomi[1].to_s+"\t"+atomi[2].to_s
+"\t"+atomj[0].to_s+"\t"+atomj[1].to_s+"\t"+atomj[2].to_s+"\t"+dist.to_s+"\n" #(dammy,x,y)
      end
    end
  end
end
```

source code

```
def find_dislocation()
  $bond=[]
  for i in 0..$atom.size-1 do
    $bond[i]=[]
  end
  for i in 0..$atom.size-1 do
    atomi=$atom[i]
    for j in 0..$atom.size-1 do
        atomj=$atom[j]
     dist = distance(atomi,atomj)
      if (dist>=$dist_min and dist<$dist_max) then # not to contain same atom
        $bond[i] << j #number i atom contain j number atom to write bond
      end
    end
    if $bond[i].size<=11 then
      $dislocation << i # $dislocation has i slipped atoms by dislocation
    end
  end
end

def red_point_dislocation()
  data=[]
  $dislocation2=[]
  file=open("3D_dislocation.tsv" ,"w")
  mk_dammy(file,4)
  for i in 0..$dislocation.size-1 do
    if $atom[$dislocation[i]][2]>$z_specify1 && $atom[$dislocation[i]][2]<$z_specify2
      file.print  0.to_s+"\t"+$atom[$dislocation[i]][0].to_s+"\t"+$atom[$dislocation[i]][1].to_s
+"\t"+$atom[$dislocation[i]][2].to_s+"\n"  ##(dammy,x,y,z)
      data.push([0.to_f,$atom[$dislocation[i]][0].to_f,$atom[$dislocation[i]][1].to_f,
$atom[$dislocation[i]][2].to_f])
      $dislocation2<<$dislocation[i]
    end
  end
end


def mk_burgers()
  teni=[]
  group=[]
  file=open("mk_burgers.tsv" ,"w")
  mk_dammy(file,4)
  for i in 0..$dislocation2.size-1 do
    count=0
    for j in 0..$dislocation2.size-1 do
      if ($atom[$dislocation2[i]][1]+($shortest*$rate) > $atom[$dislocation2[j]][1]
&& $atom[$dislocation2[i]][1]-($shortest*$rate) < $atom[$dislocation2[j]][1])
        count +=1
      end
    end
    if count >=3 then
      teni<<$dislocation2[i]# teni contain 9 atoms 3 atoms nl y_distance
    end
  end
    puts "preteni"
    pp teni
```

source code

```
  count=0
  while teni.size!=0 do
    group[count]=[]
    for j in 0..teni.size-1 do
      if ($atom[teni[0]][1]+($shortest*$rate) > $atom[teni[j]][1] &&
$atom[teni[0]][1]-($shortest*$rate) < $atom[teni[j]][1])
        group[count] << teni[j]
      end
    end
    for z in 0..group[count].size-1 do
      teni.delete(group[count][z]) # this code delete same number of group[count]
from teni's array
    end
    count=count+1
    pp group
    puts "teni"
    pp teni
  end
  nl_group=[]# array contain first nl atoms to make BGCircuit
  ncount=0
  for i in 0..group.size-1 do
    nl_group[ncount]=[]
    for j in 0..group.size-1 do
      for a in 0..$atom.size-1 do
        nl_atom=$atom[group[i][j]]
        baseatom=$atom[a]
        nz=nl_atom[2]
        if distance(nl_atom,baseatom) < $dist_max && baseatom[2] > $z_specify1
 && baseatom[2] < $z_specify2
          nl_group[ncount] << a
        end
      end
    end
    ncount+=1
  end
  for i in 0..group.size-1
    for j in 0..group.size-1
      nl_group[i].delete(group[i][j])
    end
  end
  pp nl_group
  for i in 0..nl_group.size-1
    for j in 0..nl_group[i].size-1
      file.print  0.to_s+"\t"+$atom[nl_group[i][j]][0].to_s+"\t"+$atom[nl_group[i][j]][1].to_s
+"\t"+$atom[nl_group[i][j]][2].to_s+"\n"
    end
  end
end
```

source code

```
filename="ymax.txt"
data=File.read(filename)
print("$dist_max の値を入力 >>>> ")
$dist_max = gets.chomp.to_f
print("$z_specify1 に与える値 a(0<a<0.5) を入力 >>>> ")
$z_specify1 = gets.chomp.to_f
$dist_min=0.1
$z_specify2=1.5-$z_specify1
$rate=0.2
$atom=[]
data.each do | line |
  line1=line.split("\t") #devide_space
  $atom << [ line1[0].to_f , line1[1].to_f, line1[2].to_f]
end
file=open("All_atom.tsv" ,"w")
mk_dammy(file,4)
for i in 0..$atom.size-1 do
  atom_xy=$atom[i]
  if atom_xy[2]>$z_specify1 && atom_xy[2]<$z_specify2
    file.print  0.to_s+"\t"+atom_xy[0].to_s+"\t"+atom_xy[1].to_s+"\t"+atom_xy[2].to_s+"\n"
  end
end
puts "$dist_max"
puts $dist_max
puts "$z_specify1"
puts $z_specify1
puts "$z_specify2"
puts $z_specify2
distance_bond() #write list of nl atoms for bond on "distance_bond.tsv"

$dislocation=[]
find_dislocation()#this method find the atoms of under 11
puts "$shortest"
puts $shortest
$dislocation=specify_area(0.3,0.3,0)#specify X zone
$dislocation=specify_area(1.0,1.0,1)#specify Y zone
$dislocation=specify_area(1.0,1.0,2)#specify Z zone
red_point_dislocation()#this method finds points of dislocation with red color by specify_area
puts "data"
mk_burgers()
```

A.3 Processingにおけるメソッドの集合FloatTableクラス

source code

```
class FloatTable {
  int rowCount;
  int columnCount;
  float[][] data;
  String[] rowNames;
  String[] columnNames;

  FloatTable(String filename) {
    String[] rows = loadStrings(filename);

    String[] columns = split(rows[0], TAB);
    columnNames = subset(columns, 1); // upper-left corner ignored
    scrubQuotes(columnNames);
    columnCount = columnNames.length;

    rowNames = new String[rows.length-1];
    data = new float[rows.length-1][];

    // start reading at row 1, because the first row was only the column headers
    for (int i = 1; i < rows.length; i++) {
      if (trim(rows[i]).length() == 0) {
        continue; // skip empty rows
      }
      if (rows[i].startsWith("#")) {
        continue;  // skip comment lines
      }
      // split the row on the tabs
      String[] pieces = split(rows[i], TAB);
      scrubQuotes(pieces);

      // copy row title
      rowNames[rowCount] = pieces[0];
      // copy data into the table starting at pieces[1]
      data[rowCount] = parseFloat(subset(pieces, 1));

      // increment the number of valid rows found so far
      rowCount++;
    }
    // resize the 'data' array as necessary
    data = (float[][]) subset(data, 0, rowCount);
  }

  void scrubQuotes(String[] array) {
    for (int i = 0; i < array.length; i++) {
      if (array[i].length() > 2) {
        // remove quotes at start and end, if present
        if (array[i].startsWith("\"") && array[i].endsWith("\"")) {
          array[i] = array[i].substring(1, array[i].length() - 1);
        }
      }
      // make double quotes into single quotes
      array[i] = array[i].replaceAll("\"\"", "\"");
    }
  }
```

source code

```
int getRowCount() {
   return rowCount;
 }

 String getRowName(int rowIndex) {
   return rowNames[rowIndex];
 }

 String[] getRowNames() {
   return rowNames;
 }

 // Find a row by its name, returns -1 if no row found.
 // This will return the index of the first row with this name.
 // A more efficient version of this function would put row names
 // into a Hashtable (or HashMap) that would map to an integer for the row.
 int getRowIndex(String name) {
   for (int i = 0; i < rowCount; i++) {
     if (rowNames[i].equals(name)) {
       return i;
     }
   }
   //println("No row named '" + name + "' was found");
   return -1;
 }

 // technically, this only returns the number of columns
 // in the very first row (which will be most accurate)
 int getColumnCount() {
   return columnCount;
 }

 String getColumnName(int colIndex) {
   return columnNames[colIndex];
 }

 String[] getColumnNames() {
   return columnNames;
 }

 float getFloat(int rowIndex, int col) {
   // Remove the 'training wheels' section for greater efficiency
   // It's included here to provide more useful error messages

   // begin training wheels
   if ((rowIndex < 0) || (rowIndex >= data.length)) {
     throw new RuntimeException("There is no row " + rowIndex);
   }
   if ((col < 0) || (col >= data[rowIndex].length)) {
     throw new RuntimeException("Row " + rowIndex + " does not have a column " + col);
   }
   // end training wheels

   return data[rowIndex][col];
 }

 boolean isValid(int row, int col) {
   if (row < 0) return false;
   if (row >= rowCount) return false;
   //if (col >= columnCount) return false;
   if (col >= data[row].length) return false;
   if (col < 0) return false;
   return !Float.isNaN(data[row][col]);
 }
```

source code

```
float getColumnMin(int col) {
   float m = Float.MAX_VALUE;
   for (int i = 0; i < rowCount; i++) {
     if (!Float.isNaN(data[i][col])) {
       if (data[i][col] < m) {
         m = data[i][col];
       }
     }
   }
   return m;
 }

 float getColumnMax(int col) {
   float m = -Float.MAX_VALUE;
   for (int i = 0; i < rowCount; i++) {
     if (isValid(i, col)) {
       if (data[i][col] > m) {
         m = data[i][col];
       }
     }
   }
   return m;
 }

 float getRowMin(int row) {
   float m = Float.MAX_VALUE;
   for (int i = 0; i < columnCount; i++) {
     if (isValid(row, i)) {
       if (data[row][i] < m) {
         m = data[row][i];
       }
     }
   }
   return m;
 }

 float getRowMax(int row) {
   float m = -Float.MAX_VALUE;
   for (int i = 1; i < columnCount; i++) {
     if (!Float.isNaN(data[row][i])) {
       if (data[row][i] > m) {
         m = data[row][i];
       }
     }
   }
   return m;
 }

 float getTableMin() {
   float m = Float.MAX_VALUE;
   for (int i = 0; i < rowCount; i++) {
     for (int j = 0; j < columnCount; j++) {
       if (isValid(i, j)) {
         if (data[i][j] < m) {
           m = data[i][j];
         }
       }
     }
   }
   return m;
 }
```

source code

```
  float getTableMax() {
    float m = -Float.MAX_VALUE;
    for (int i = 0; i < rowCount; i++) {
      for (int j = 0; j < columnCount; j++) {
        if (isValid(i, j)) {
          if (data[i][j] > m) {
            m = data[i][j];
          }
        }
      }
     }
    return m;
  }
}
```

A.4　転位視覚化プログラム

source code

```
FloatTable data;
float[][] data_bond = new float[10000][7];//distance_bond.tsv_gyou
float[][] data_dis = new float[10000][3];//3D_dislocation.tsv_gyou
float[][] data_atom = new float[10000][3];//All_atom.tsv_gyou
float[][] data_bur = new float[10000][3];//mb_atom.tsv_gyou
int a,b,R1,B1,R2,B2,B_size,mvX=0, mvY=0, oldX=0, oldY=0, nowX=0, nowY=0;
float l,z_max,z_min,nz1,nz2;
float depth=0.0;
float size=100;
int pitch=50;
float zoom=1.0;

void setup() {
  size (1200, 700, P3D);
  colorMode(RGB, 100);
  background(90);
 // String loadpath = selectInput();

  data = new FloatTable("distance_bond.tsv");
  int d_bond = data.getRowCount();
  for (int row = 0; row < d_bond; row++) {//tsv_gyou
      for (int col = 0; col < 6; col++) {
  data_bond[row][col] = size*data.getFloat(row, col);
  // print(data_bond[row][col]);
      }
  }
  data = new FloatTable("3D_dislocation.tsv");
  int d_dis = data.getRowCount();
  for (int row = 0; row < d_dis; row++) {//tsv_gyou
      for (int col = 0; col < 3; col++) {
  data_dis[row][col] = size*data.getFloat(row, col);
  //println(data_dis[row][col]);
      }
  }
  data = new FloatTable("All_atom.tsv");
  int atom = data.getRowCount();
  for (int row = 0; row < atom; row++) {//tsv_gyou
      for (int col = 0; col < 3; col++) {
  data_atom[row][col] = size*data.getFloat(row, col);
      }
  }
    data = new FloatTable("mk_burgers.tsv");
    //data = new FloatTable("nonbur.tsv");
  int burgers = data.getRowCount();
  for (int row = 0; row < burgers; row++){//tsv_gyou
      for (int col = 0; col < 3; col++) {
  data_bur[row][col] = size*data.getFloat(row, col);
      }
  }
}
void drawAxis(float length) {
  stroke(0, 100, 0);
  line(0, 0, 0, length, 0, 0);
  stroke(0, 100, 0);
  line(0, 0, 0, 0, 0, -length);
}
```

source code

```
void draw() {
  lights();
  scale(1);
  background(90);

  if (mousePressed) {
    mvX=mouseX-oldX;
    mvY=mouseY-oldY;
    oldX=mouseX;
    oldY=mouseY;
  }
  else {
    mvX=0;
    mvY=0;
    oldX=mouseX;
    oldY=mouseY;
  }
  nowX += mvX;
  nowY -= mvY;
  if(keyPressed){
    if(key=='p'){
      depth+=10;
    }else if(key=='n'){
      depth-=10;
    }
  }
  translate(width/2, height/2, depth);

  rotateX(radians(nowY/3));
  rotateY(radians(nowX/3));
  if(keyPressed){
    if(key=='a'){
       zoom+=0.1;
    }else if(key=='b'){
      zoom-=0.1;
    }
  }

  translate(0,3.615569/2*size*zoom,1.5/2*size*zoom);
  //write bond between atom1 and neibor atom2
  stroke(0, 0, 0);
  strokeWeight(3);
  data = new FloatTable("distance_bond.tsv");
  int d_bond = data.getRowCount();
  for (int row = 0; row < d_bond; row++) {//tsv_gyou
      line(data_bond[row][0]*zoom, -data_bond[row][1]*zoom, -data_bond[row][2]*zoom,
   data_bond[row][3]*zoom, -data_bond[row][4]*zoom, -data_bond[row][5]*zoom);
      fill(240, 100, 100);
  }

  data = new FloatTable("All_atom.tsv");
  int atom = data.getRowCount();
  z_max=data_atom[0][2];
  z_min=data_atom[0][2];
  for (int row = 0; row < atom; row++) {//tsv_gyou
      if (z_max<data_atom[row][2]){
  z_max=data_atom[row][2];}
      if (z_min>data_atom[row][2]){
  z_min=data_atom[row][2];}
  }
  l=z_max-z_min;
  //println(l);
```

source code

```
  //draw all atoms by red or blue color
  for (int row = 0; row < atom; row++) {//tsv_gyou
      nz1 = data_atom[row][2];
      R1=int(255*(1-(nz1-z_min)/l));
      B1=int(255*(nz1-z_min)/l);
      strokeWeight(12);
      stroke(R1,0,B1);
      point(data_atom[row][0]*zoom, -data_atom[row][1]*zoom, -data_atom[row][2]*zoom);
      fill(240, 100, 100);
  }
  // draw atoms of nl under 11 by red & blue color
  data = new FloatTable("3D_dislocation.tsv");
  int d_dis = data.getRowCount();
  for (int row = 0; row < d_dis; row++) {//tsv_gyou
      nz2 = data_dis[row][2];
      R2=int(255*(1-(nz2-z_min)/l));
      B2=int(255*(nz2-z_min)/l);
      strokeWeight(20);
      stroke(R2,0,B2);
      point(data_dis[row][0]*zoom, -data_dis[row][1]*zoom,  -data_dis[row][2]*zoom);
      fill(240, 100, 100);
  }
    data = new FloatTable("mk_burgers.tsv");
    // data = new FloatTable("nonbur.tsv");
  int burgers = data.getRowCount();
  for (int row = 0; row < burgers; row++){//tsv_gyou
      strokeWeight(20);
      stroke(0,255,0);
      point(data_bur[row][0]*zoom, -data_bur[row][1]*zoom, -data_bur[row][2]*zoom);
      fill(240, 100, 100);
  }
}
```