卒業論文

らせん転位のMayaによる視覚化

関西学院大学　理工学部　情報科学科
5715　井原　航

2009年　3月

指導教員　　西谷　滋人　教授

概 要

らせん転位は結晶中に現れる欠陥の一つである．結晶とは原子または分子が規則正しく配列された固体である，しかし，実際の結晶はいたる所に配列の乱れが生じ，様々な欠陥が存在する．らせん転位は不明な点が多く，特に中心付近の構造が明らかになっていない．現在，次世代半導体素子材料として，SiC が注目されている．SiC は積層周期の違いから 200 種類もの結晶多形が存在する．その中でも主要なものとして，4H-SiC，6H-SiC，3C-SiC の 3 種類が挙げられる．その SiC の結晶中でも，実際にらせん転位は確認されており，中心のマイクロパイプと呼ばれる欠陥は大電力デバイスにとって致命的な問題となっている．本研究では，作成するモデルの対象を 4H-SiC とし，らせん転位を視覚化する．視覚化することにより，肉眼では見ることができない原子レベルの構造を確認することができる．モデルの表示方法として，Stick&ball 表示法と四面体表示法を用いた．それぞれの表示方法で様々な角度からのモデル作成を行った．それにより，中心付近の乱れた結合の様子がわかった．また，藤井（西谷研究室）が行った SiC の主要な面の表面エネルギーの計算とマイクロパイプ部分の面とを照らし合わせることによって SiC の気相成長中に生成するマイクロパイプの構造に対して原子サイズのモデルを構築した．

視覚化は，Maya という 3DCG 作成ソフトを用いて行った．Maya を制御する MEL と呼ばれるスクリプト言語があるが，ループや線形計算の記述が複雑になっている．そこで，シンプルな言語である Ruby を用いて MEL を出力させ．格子モデルを作成した．

目 次

第1章　序論

1.1　らせん転位

らせん転位は結晶中に現れる欠陥の一つである．らせん転位の大きな特徴は，図 1.1 のように転位線を中心に結晶面を一周すると元の位置には戻らずユニットセル分ずれた場所にくるという点である．積層がずれたことによってできた段の部分をステップと呼び，ここに原子が取り込まれていき，らせん状に結晶は成長していく．正常な原子と違い，ステップが消えることなく結晶が成長し続けるので単結晶を生成しやすいという利点がある．しかし，らせん転位は積層のずれによる原子結合のひずみや原子種の違いなど，明確でない点が多く存在する．実際に，SiC の結晶でらせん転位は確認されており，SiC の結晶成長に大きな影響を及ぼすと考えられている．

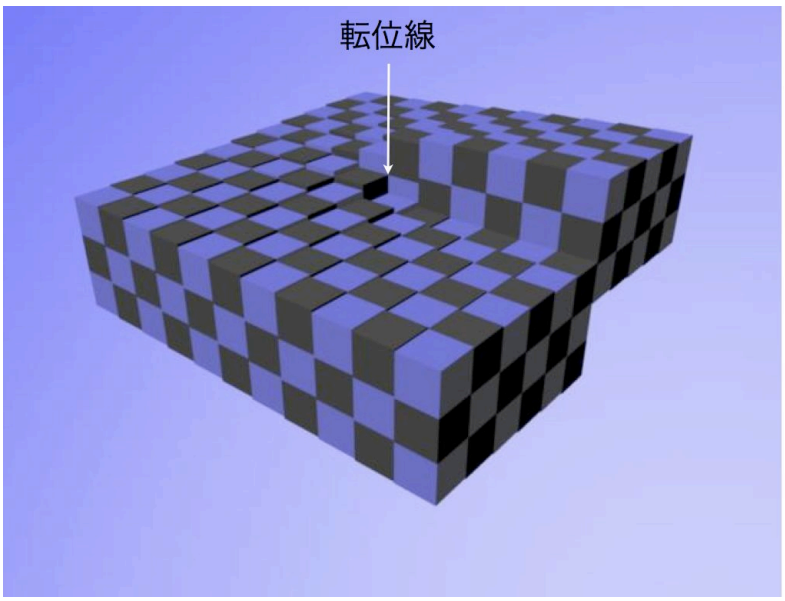

図 1.1: らせん転位模式図.

1.2 SiC

現在，最も多く用いられているパワーエレクトロニクス半導体材料は SiC であるが，Si はデバイス性能が理論限界に近づいており，これ以上の改善が期待できない現状にある．そこで，Si よりも耐熱性，オン抵抗に優れた SiC が注目されている．SiC はワイドギャップ半導体と言われ，荷電帯から伝導帯に電子を持ち上げるのに必要なエネルギーが Si に比べ，約 3 倍必要である．このハンドギャップにより，Si に比べて SiC の絶縁破壊電解度は 10 倍にもなる．これにより，同じ耐圧を得るのに，厚みが 1/10 にできる．また，SiC は Si に比べて融点が高く，Si が 200 ℃以下であるのに対し，SiC は 600 ℃程度まで動作可能であり，耐熱性にも優れている [1]．いずれもパワーデバイスに要求される特性であり，実用化が期待されている．SiC は積層周期の違いから 200 種類もの結晶多形が存在する．その中でも主要なものは六方晶の 4H，6H 及び立方晶の 3C の 3 種類である．

1.3 視覚化

視覚化によって，人間が直接見ることができないものを，見ることができるようになる．本研究において，人間が直接見ることができないものは原子レベルでの SiC の構造である．視覚化を行うと，理論や数値だけでは理解しにくいことでも，実際に目で確認できるので容易に理解することができる．らせん転位の中心付近での第一原理計算は不可能であり，具体的なことは何もわかっていない．本研究では，4H-SiC を対象とした，格子モデル作成によるらせん転位の視覚化を行う．様々なモデルを作成することにより，らせんの中心ではどのような構造をとっているのかについて解析し，また，らせん転位が現れた結晶はどのような結晶成長の機構を持つのか予測することを目的とする．

第2章　手法

モデルの作成はMayaという3DCG作成ソフトによって行った．Mayaは独自に開発されたMELと呼ばれるスクリプト言語によってすべての機能をコントロールしたり，カスタマイズや拡張などを自由に行うことができる[2]．しかし，MELは一般的な言語に比べてループや線形計算の記述が複雑になっている．そこでシンプルなスクリプト言語であるRuby[3]で作成したプログラムによってMELを出力させた．

2.1　Maya, MEL

一口に「CG」といっても，その種類は多岐にわたっている．Adobe Photoshopや，Adobe Illustratorなどのフォトレたち，ベクターデザイン系のソフトで作った絵もCGの一種である．工業デザインの分野で使われている，いわゆるCADと呼ばれるものもCGに含まれるといっていい．そんな多様性を持つCGというジャンルにおいて，Mayaは主に映画やゲームといったエンターテイメント分野向けのCG，とりわけフォトリアリスティックな「3DCGアニメーション」の作成を目的として，カナダのエイリアスシステムズ（現在のオートディスクと合併）が開発したソフトである．MELという独自の言語を用いて，Mayaを自分の作成スタイルに合わせて自在にカスタマイズすることができる．

2.2　Ruby

Rubyはスクリプト言語である．CやJavaの様なプログラム言語で書かれたプログラムを実行するには，そのプログラムのソースコードを機会命令に翻訳する「コンパイル」という作業が必要になる.「スクリプト言語」の「スクリプト」という言葉は,「プログラム」とほとんど同じような意味で使われているが，スクリプト言語の場合，書いたスクリプトはコンパイルする必要がなく，そのまま実行できる．そのため，コンパイルの必要な言語に比べ，プログラミングは手軽である．また，Rubyはオープンソースソフトウェア（フリーソフト）である．1995年に，インターネット上で公開され誰もがRubyを入手して，自由に使用することができる．

第3章　結果

3.1　Stick & Ball表示

原子レベルの視覚化に適したStick & Ball表示法を用いた．これは原子をballで，原子間の結合をStickで表す手法である．

3.1.1　安定状態のモデル

図3.1は4H-SiCの安定状態のモデルである．まず，ユニットセルのデータを読み込ませ，スーパーセルに拡張したのち表示させている．このモデルを作成するプログラムはすでに開発されていたので利用し研究をすすめた．

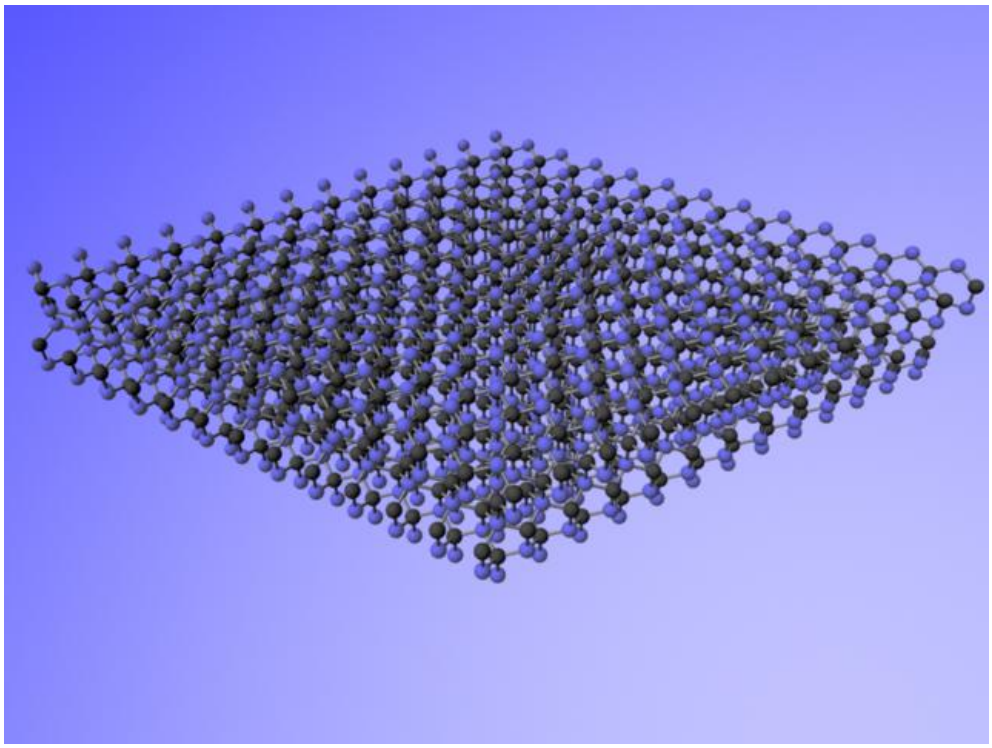

図 3.1: 安定状態原子モデル

3.1.2 spiral 関数と中間ファイル

らせん転位モデルを作成するための積層をずらすプログラムを示す.

```
def make_spiral(ball1,burgers,r0)
  sp_ball = []
  for j in 0..ball1.size-1 do
    x = ball1[j][0]
    y = ball1[j][1]
    z = ball1[j][2]

    tmp = Vector[x,0,z]

    r = tmp.r
    if(r>r0)then
      theta = asin(z/r)
      t = theta*(180/PI)
      if(x<0)then t = 180-t end
      if(t<0)then t = 360+t end

      xx = r*cos(t*(PI/180))
      yy = y+(burgers*(t/360))
      zz = r*sin(t*(PI/180))

      sp_ball.push Vector[xx,yy,zz]
    else
      sp_ball.push Vector[x,y,z]
    end
  end
  return sp_ball
end
```

引数として ball1(ずらす前の原子位置)，burgers(ずれる面の大きさと方向を表すバーガーズベクトル)，r0(ホローコアの半径) を与えている．積層をユニットセル分ずらした後の原子位置をデータとして残し，それを中間ファイルとした．以下にその一部を示す.

1 行目はファイル名，3 行から 5 行は 4H-SiC のユニットセルのプリミティブベクトルである．

```
4H-SiC_spiral
    1.000000000000000
     3.0800000000000001      0.0000000000000000      0.0000000000000000
     -1.5399999300000000      2.6673582799999997      0.0000000000000000
     0.0000001800000000      0.0000003200000000      10.0809999999999995
```

Direct の後にある 2 つの数字は左から Si の原子数，C の原子数である．その下にある数字は 1 行ずつ原子の座標を表しており，Si，C の原子数の数だけ上から順に配列に入る．つまり，座標の上 4 行は Si の座標で，下 4 行は C の座標である．尚，この 8 行にわたる座標は 4H-SiC のユニットセルの座標である．

```
Direct  4  4
  0.0000000000000000  0.0000000000000000  0.0000000000000000
  0.0000000000000000  0.0000000000000000  0.5000000000000000
  0.3333333300000021  0.6666666699999979  0.2500000000000000
  0.6666666699999979  0.3333333300000021  0.7500000000000000
  0.0000000000000000  0.0000000000000000  0.1875000000000000
  0.0000000000000000  0.0000000000000000  0.6875000000000000
  0.3333333300000021  0.6666666699999979  0.4375000000000000
  0.6666666699999979  0.3333333300000021  0.9375000000000000
```

同じように以下の数字も Si，C の座標である．こちらはらせん転位の座標である．Direct の数字は 125 と 100 だが，多いので途中抜いている．

```
Direct  125  100
  -3.0800001399999997  -5.3347165599999995  6.7206666445536127
  -3.0800000500000042  -5.3347163999999969  11.7611666440176563
#  -3.0799999600000039  -5.3347162399999970  16.8016666434817026
#  -4.6199998900000008  -2.6673579599999999  15.9615832759934424

  -3.0800001062500004  -5.3347164999999999  8.6108541443526292
  -3.0800000162500001  -5.3347163399999999  13.6513541438166737
  -3.0800000299833221  -3.5564775577754779  10.8260588269102023
  -1.5399999325166756  -4.4455968422245213  16.4766494722837322
```

3 つ目と 4 つ目の座標の左にある#は，ファイルを読み込ます際に，#のある行の原

子を省いて読み込ます時に使う．こうすることによって，表示を避けたい原子などを，ファイルには残したまま表示させないようにすることができる．また，ファイル自体に直接入力したり，削除したりしても対応可能なので，自由な表示を実現することができる．さらに，一度中間ファイルを残しておくことで，モデル作成のための処理の高速化も図れる．

3.1.3　らせん転位モデル

図 3.2 は作成したらせん転位のモデルである．この図だけでは一見わからないが，らせんの中心で原子結合がきわめて乱れた状態となっていることが示されている．

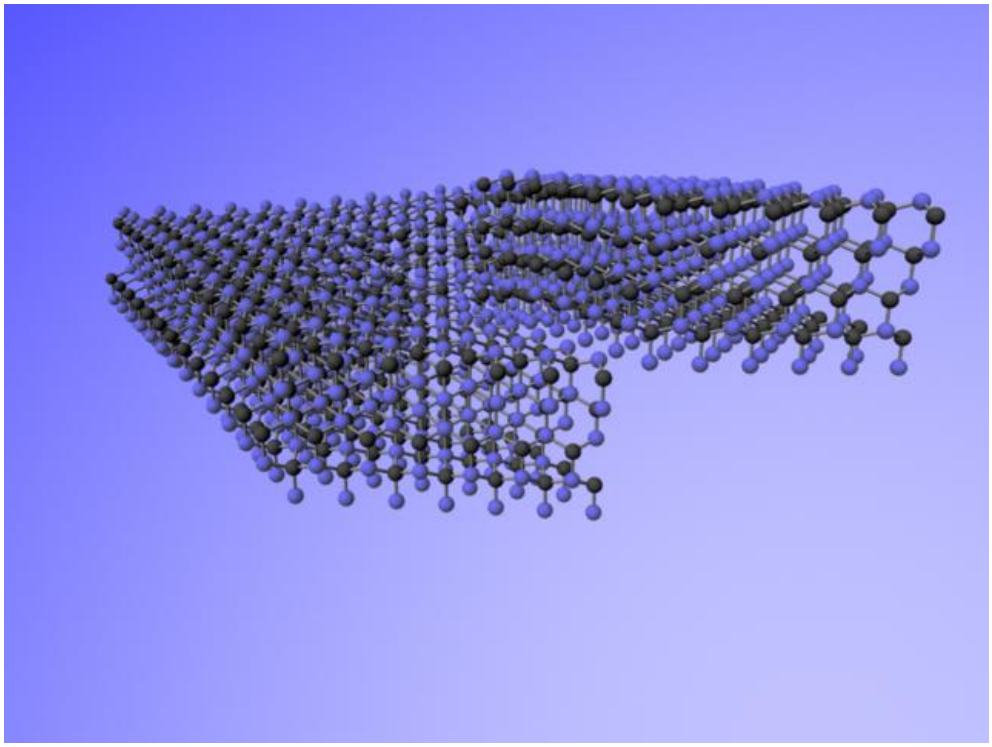

図 3.2: らせん転位原子モデル

3.2　四面体表示

4H-SiC は 1 つの Si 原子に対して 4 つの C 原子がまわりに存在し，Si 原子を中心，C 原子を頂点とする四面体を構成することができる．また，その逆 (中心 C 原子，頂点 Si 原子) も可能である．Stick&ball 表示では一目でどこに異常があるのか見つけるのは困難だった．しかし，四面体表示にすることによって異常がある場所が一目で明らかになる．

3.2.1　make tetra 関数

図 3.3 は青の ball を中心，黒の ball を頂点とした四面体を示している．この表示法を 4H-SiC に適用したものが図 3.4 である．安定状態の 4H-SiC は原子が規則正しく配置されているので四面体もきれいに構成されていることがわかる．

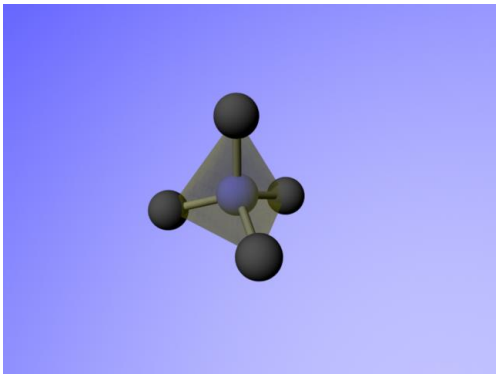

図 3.3: 四面体

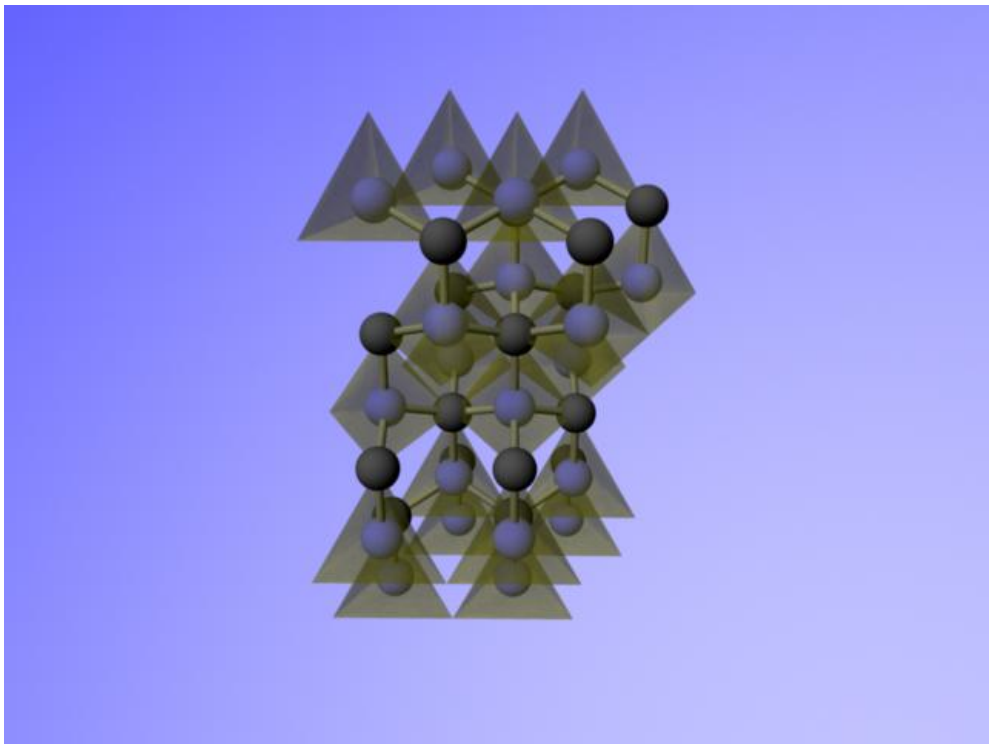

図 3.4: 4H-SiC 四面体表示

中心にしたい原子の座標を与え，最近接の原子 4 つで四面体を構成する関数を作成した．以下にそのプログラムの一部を示す．

1 つの四面体を作成する関数である．tetra_spiral_cone_number は今までにいくつの四面体の面となる三角形が作成されたかカウントする変数である．MEL スクリプトを出力する時に，1 つ 1 つの三角形を区別するための番号となるので重要な役割がある．引数として file1(出力されるファイル)，ball(頂点)，center(中心)，len(頂点までの距離
の範囲) を与えている．

```
$tetra_spiral_cone_number = 0
def tetra_spiral(file1,ball,center,len)
  ball1 = []
  ball2 = []
  ball3 = []
  count = 0
  s = 0
```

MEL を出力するスプリクト．三角形を作る関数である．

```
file1.printf("global proc tetrahedron")
file1.printf("(float $x,float $y,float $z,")
file1.printf("float $sx,float $sz)\n{\n")
file1.printf("cone;\n")
file1.printf("move $x $y $z;\n")
file1.printf("scale $sx 0.0 $sz;\n")
file1.printf("sets -e -forceElement %sSG;\n}\n",tetra_color)
```

与えられた中心に対して，すべての原子との距離を計算し，len の範囲内ならば頂点の候補として配列に座標を格納する．その時，その原子が Si か C かを判断するための値も同時に格納する．5 つ目の候補が現れた場合は，それまでに格納された 4 つの原子と合わせた 5 つの頂点の候補から，最も中心から離れている原子を配列から削除する．こうして，頂点の候補の配列には常に 4 つの座標しかない状態を維持する．

```
tetra = []
tetra_tmp = []
delete_num = []
for ii in 0..ball.size-1 do # 四面体の頂点のループ
  for jj in 0..ball['atom'+ii.to_s].size-1 do
    tmp = ball['atom'+ii.to_s][jj] - center
    if(len[0]<tmp.r && tmp.r<=len[1])then
      tetra.push [ii,ball['atom'+ii.to_s][jj]]
      if(tetra.size>4)then
        max = 0
        len_array = []
        tetra.each do |vec|
          t_tmp = vec[1] - center
          if(t_tmp.r>max)then max = t_tmp.r end
          len_array.push t_tmp.r
        end
        head = len_array.index(max)
        tetra.delete_at(head)
      end
    end
  end
```

```
  end
```

もし，ここまでの計算で，頂点の候補が 4 つ以上みつからなかった場合は，四面体を作成するのは不可能なので終了する．

```
  if(tetra.size!=4)then
    file1.printf("n")
    return file1
  end
```

頂点の候補が 4 つある場合は四面体作成のための計算に入る．Maya で図形を作る時，最初に表示される位置は原点が基準になる．そこから，移動させる位置や回転させる角度を与えて図形の場所を動かすので，その位置情報を計算するのである．まず，4 つの面の頂点の座標を新しく配列に格納する．面それぞれに対して，重心，法線ベクトル，重心から各頂点への方向ベクトルを計算する．それらを vec_data として保存する． 重心は三角形の位置，法線ベクトルは面を合わせるため，重心から各頂点への方向ベクトルは頂点を合わせるために用いる．

```
  if(tetra.size>=4)then
    for i in 0..3 do
      if(tetra_tmp[i]>key_length+0.2)then
        ball1.push tetra[i][1]
        count = 1
      end
      if(tetra_tmp[i]<key_length-0.2)then
        ball2.push tetra[i][1]
        count = 2
      end
    end

    tri = [[tetra[0][1],tetra[1][1],tetra[2][1]], # 各面の三角形の頂点
           [tetra[1][1],tetra[2][1],tetra[3][1]],
           [tetra[2][1],tetra[3][1],tetra[0][1]],
           [tetra[3][1],tetra[0][1],tetra[1][1]]]

    vec_data = [] # 重心の座標　法線ベクトル 方向ベクトル
    for k in 0..tri.size-1 do
```

```
    x = (tri[k][0][0] + tri[k][1][0] + tri[k][2][0])/3
    y = (tri[k][0][1] + tri[k][1][1] + tri[k][2][1])/3
    z = (tri[k][0][2] + tri[k][1][2] + tri[k][2][2])/3
    median = Vector[x,y,z] # 重心

    dir_nor = median - center # 四面体重心から三角形重心への方向
ベクトル

    a = tri[k][1] - tri[k][0] #  平面ベクトル1
    b = tri[k][2] - tri[k][0] #  平面ベクトル2
    x = a[1]*b[2]-a[2]*b[1]   # ---------
    y = a[2]*b[0]-a[0]*b[2]   # 外積計算
    z = a[0]*b[1]-a[1]*b[0]   # ---------
    normal_vec = Vector[x,y,z] # 法線ベクトル
    if(dir_nor[0]*normal_vec[0]<0 ||
       dir_nor[1]*normal_vec[1]<0 ||
       dir_nor[2]*normal_vec[2]<0)then
      normal_vec = normal_vec*(-1)
    end
    directional_vec = tri[k][0] - median
    # 面重心から基準の頂点への方向ベクトル

    vec_data.push [median,normal_vec,directional_vec]
  end


  # 以下の計算は Rotate_Matrix.mw を参考
  for k in 0..vec_data.size-1 do
    flag = 0
    x = vec_data[k][1][0]
    y = vec_data[k][1][1]
    z = vec_data[k][1][2]
    len = vec_data[k][1].r
    if(len==0)then next end
    theta1 = asin(z/len)

    tmp = x/(len*cos(theta1))
    if(-1.001<=tmp && tmp<-0.999)then
      theta3 = 1.5707963267949
```

```
elsif(0.999<=tmp && tmp<1.001)then
  theta3 = -1.5707963267949
else
  theta3 = -asin(tmp)
end

if(y<0)then
  theta1 = theta1*-1
  theta3 = theta3*-1
end
xx = theta1*180/PI
zz = theta3*180/PI

ro_x = Vector[cos(theta3),sin(theta3),0]
#回転後の x 軸の向き
ro_z = Vector[sin(theta3)*sin(theta1),
              (-1)*cos(theta3)*sin(theta1),
              cos(theta1)]
#回転後の z 軸の向き
dir = (vec_data[k][2])*(1/vec_data[k][2].r)
# 重心から基準の頂点への方向ベクトル
flag = 0
#  dir ベクトルが xz 平面上の第 n 象限にいるか判定
kk = [1,-1]
array = []
for iii in 0..kk.size-1 do
  tx = ro_x*kk[iii]
  tz = ro_z
  lx = tx.r
  lz = ro_z.r
  ld = dir.r
  inner = (dir[0]*tx[0]+dir[1]*tx[1]+dir[2]*tx[2]) # 内積
  if(-1.001<=inner && inner<-0.999)then
    thetaX = 3.141592654
  elsif(0.999<=inner && inner<1.001)then
    thetaX = 0.0
  else
    thetaX = acos(inner/(lx*ld))
  end
```

```
  #thetaX = acos(inner/(lx*ld))
  if(iii==0)then theta2 = thetaX end
  inner = dir[0]*tz[0]+dir[1]*tz[1]+dir[2]*tz[2] # 内積
  if(-1.001<=inner && inner<-0.999)then
    thetaZ = 3.141592654
  elsif(0.999<=inner && inner<1.001)then
    thetaZ = 0.0
  else
    thetaZ = acos(inner/(lz*ld))
  end
  gg = (thetaX+thetaZ)*180/PI
  if(gg<=91 && gg>-1)then
    flag = 1
  end
end
if(flag==1)then #１，２象限にいる場合，回転方向を逆向きにする
  theta2 = theta2*-1
end
yy = theta2*180/PI


tmp_sub = (tri[k][1] + tri[k][2])*0.5
tmp1 = (tri[k][0] + tmp_sub)*0.5
tmp2 = (tri[k][1] - tri[k][2])*0.5
```

計算された情報を先程作成した関数に当てはめ，四面体完成である．

```
$tetra_spiral2_cone_number += 1
cc = $tetra_spiral2_cone_number
if(count == 0)then
  file1.printf("tetrahedron(%7.9f,%7.9f,%7.9f,"
               ,tmp1[0],tmp1[1],tmp1[2])
  file1.printf("%7.9f,%7.9f);\n"
               ,(tmp1-tmp_sub).r,tmp2.r)
  file1.printf("rotate -r -os %3.9f 0 %3.9f nurbsCone%d ;\n"
               ,xx,zz,cc)
  file1.printf("rotate -r -os 0 %3.9f 0 nurbsCone%d ;\n"
               ,yy,cc)
end
```

```
      end
    end

    return file1
  end
```

3.2.2 異常抽出

図 3.5 は Stick&ball と四面体を合わせて表示させたものである．安定状態の四面体の中心から頂点までの距離を基準として，それよりも約 10%以上離れてしまっている頂点を含んでいる四面体を異常であると判定し，赤色の四面体を構成させている．異常判定の基準だが，この距離よりも短くしてしまうと，赤色の四面体は表示されなくなり，長くすると赤色の方が多くなってしまうので，適当だと言える．こうして見ると中心に異常があるとよくわかる．また，中心から離れた所では正常な四面体が構成されているので異常がないこともわかる．

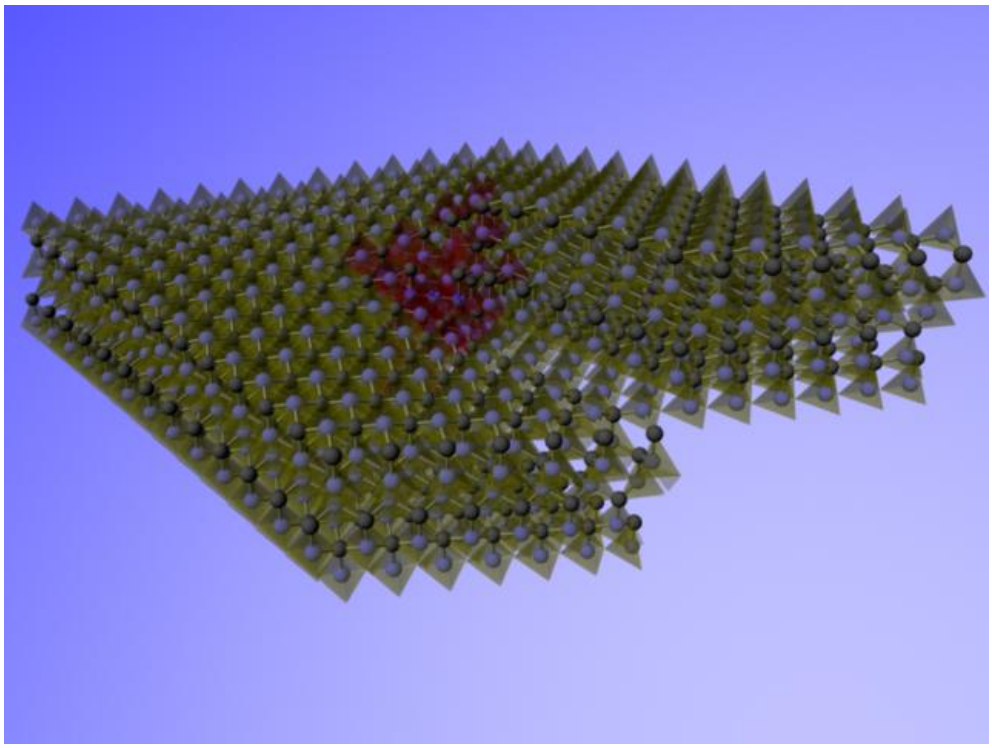

図 3.5: Stick&Ball 四面体表示

図 3.5 から Stick&Ball を取り除き，四面体だけにしたものが図 3.6 である．正常四面体であれば，Si 原子が中心なら頂点は C 原子，C 原子が中心なら頂点は Si 原子，というように四面体の中心と頂点は異なる原子で構成される．しかし，らせん転位のモデルには頂点に中心と同じ種類の原子を含んでいる四面体が存在する．それが図 3.5 の中心部に見られる ball と stick である．この stick は原子の結合を表しているのではなく，中心と頂点の関係にある ball 同士をつないでいる．このようなことが起こる理由として，積層がずれることによって中心部の原子が押し上げられ，原子が近づいてしまったからだと思われる．

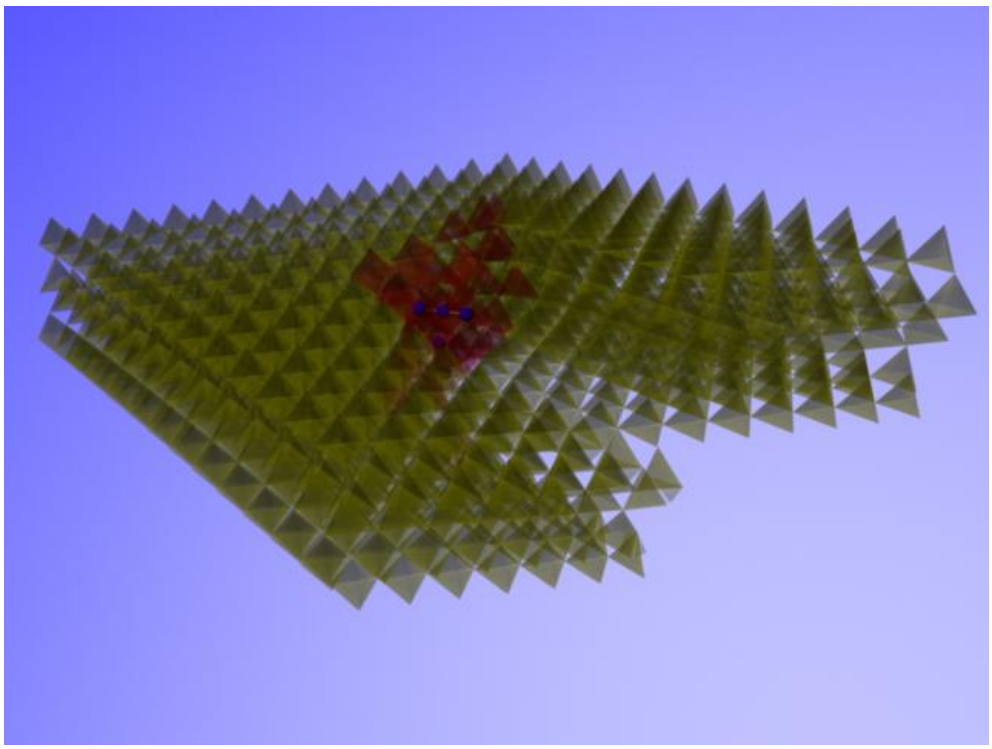

図 3.6: 四面体表示モデル

3.3　中心の異常解析

四面体で表示することにより，らせんの中心で異常が起こっていることがわかった．中心の異常をさらに見やすく表示させ，中心の様子を解析していく．

3.3.1　中心部くりぬき

図 3.7 はらせんの中心部のみくりぬいた図である．中心から離れた，異常のない部分を削除することにより中心部を見やすくすることができた．図 3.8 は，さらに異常な四面体のみを表示した図である．

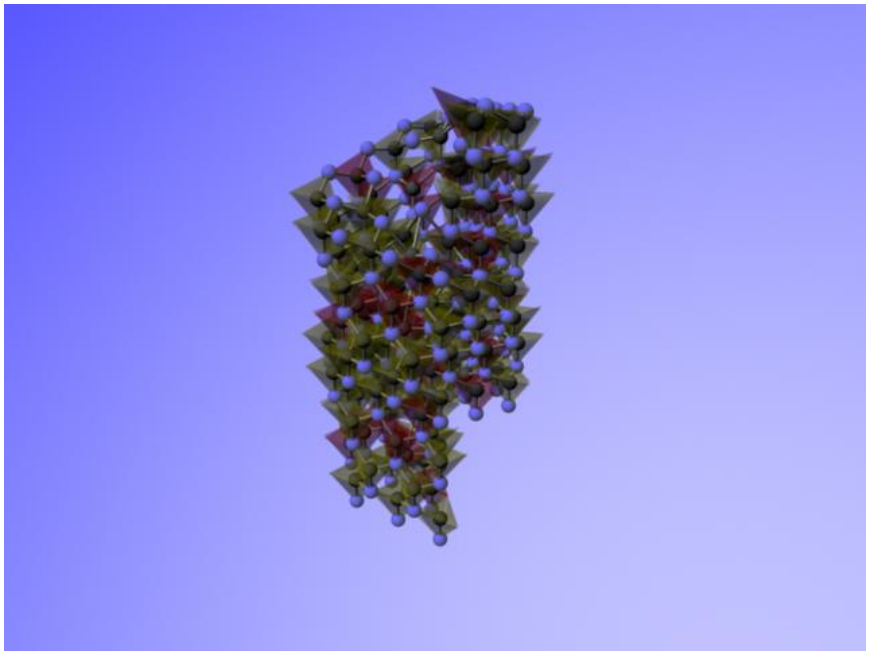

図 3.7: 中心部くりぬき

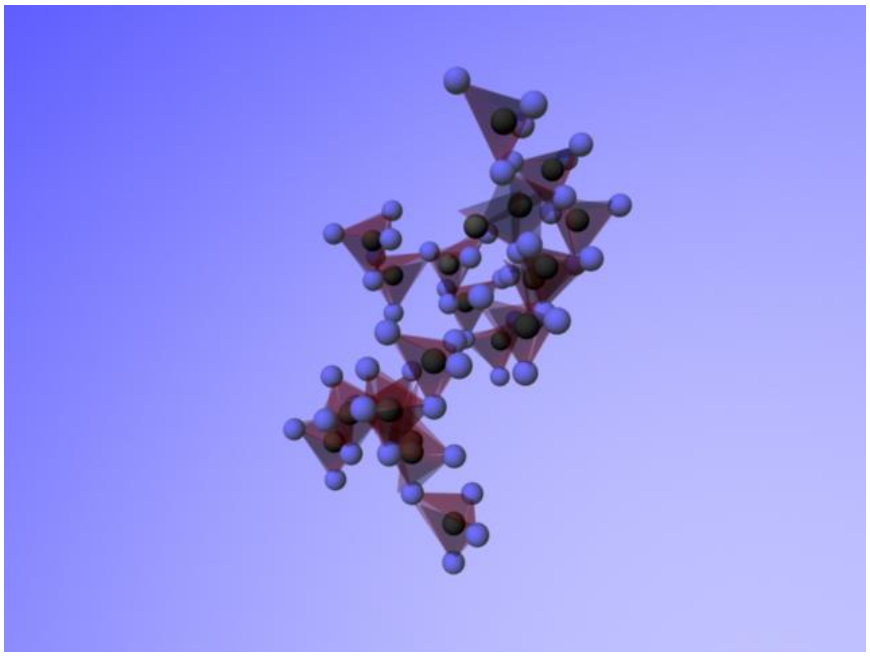

図 3.8: 異常判定された四面体

3.3.2　中心位置変更

今まで作成してきたモデルは，すべてらせんの中心の位置は図 3.9 の 1 の場所であった．しかし実際に転位は常に同じ位置にらせんの中心がくるわけではない．そこで中心の位置を 2 の場所に変更した場合，1 の時と比べてどのような変化が起こるか調べた．

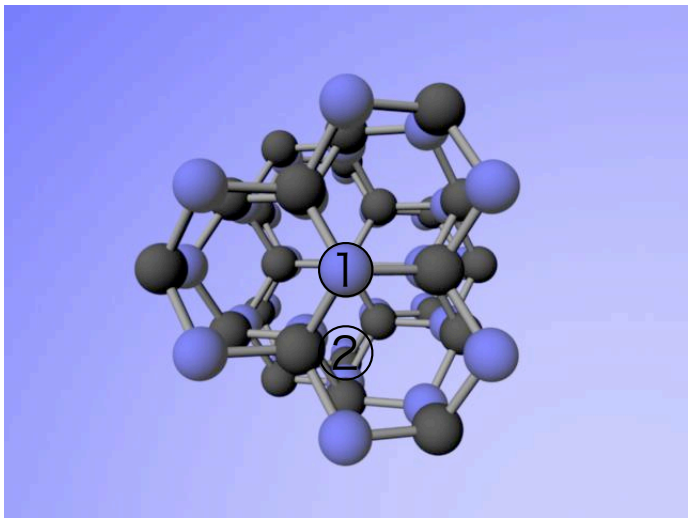

図 3.9: 中心位置

図 3.10 は図 3.9 の 2 の点に中心を変更させたモデルである．

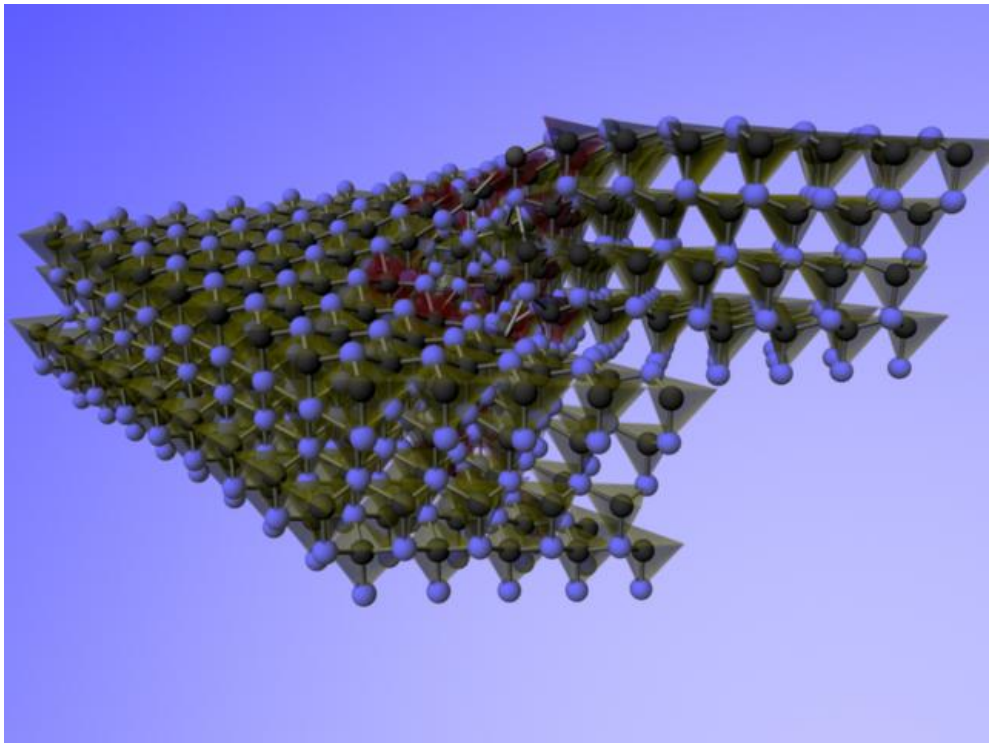

図 3.10: 中心位置変更後

図 3.11，図 3.12 は，それぞれ中心位置を変更した前後の中心部を上から見たものである．両方とも図の中心から真っすぐ右にすべり面が存在している．中心の位置を変更すると異常と判定された四面体の分布も変わっており，変更前と比べると変更後の方がその数も多くなっている．

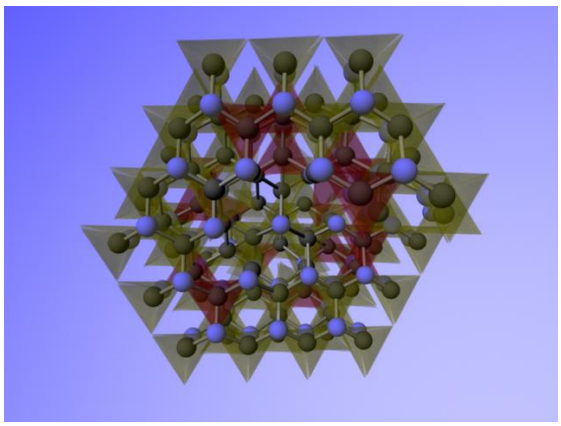

図 3.11: 中心位置変更前上部

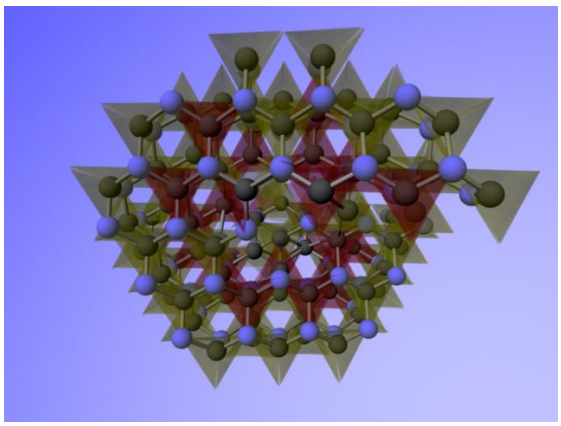

図 3.12: 中心位置変更後上部

図 3.13 は変更後の異常判定された四面体のみを表示させたものである．こちらを見ると四面体の数が図 3.8 よりも多くなっていることがよく分かる．

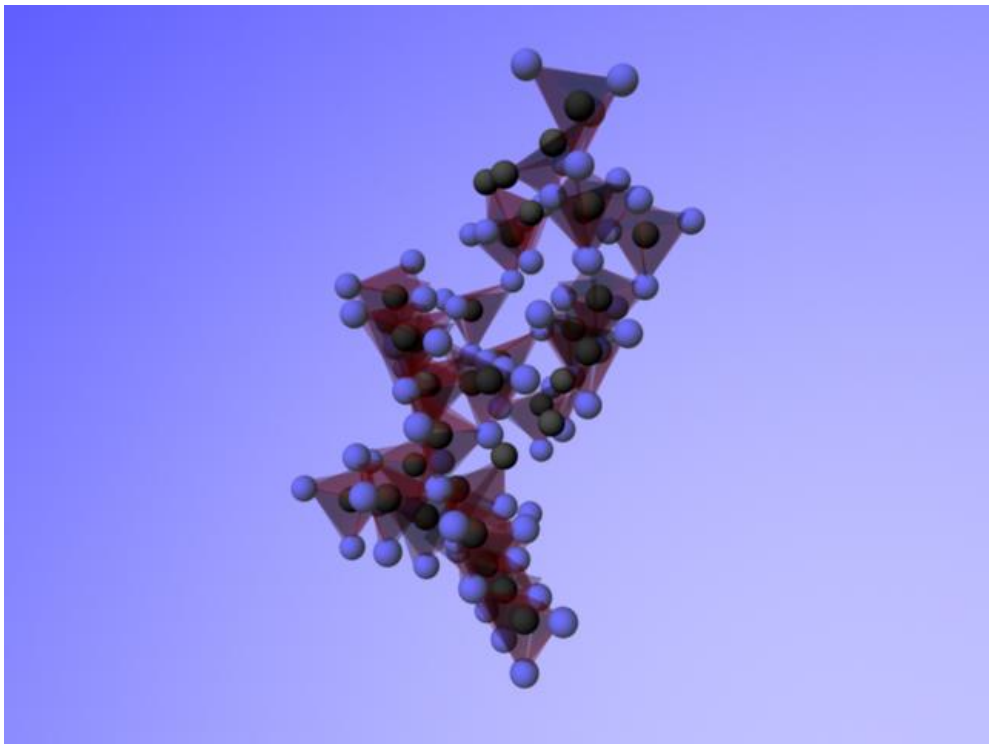

図 3.13: 異常判定された四面体 中心位置変更後

また，中心位置変更後のモデルについては，変更前には見られなかった異常が確認できた．

3.3.3　中心付近の異常原子

図 3.14，図 3.14 はらせんの中心変更後の中心付近の様子を拡大したものである．中心にある C 原子 (黒の ball) の結合の仕方や，ボンドの数がおかしくなってしまっている．これは，積層がずれることによって，本来ならあり得ない場所に Si 原子 (青の ball) が配置されたからだと考えられる． この異常のように，四面体表示だけでは抽出しきれない異常が他にもあるかもしれない．

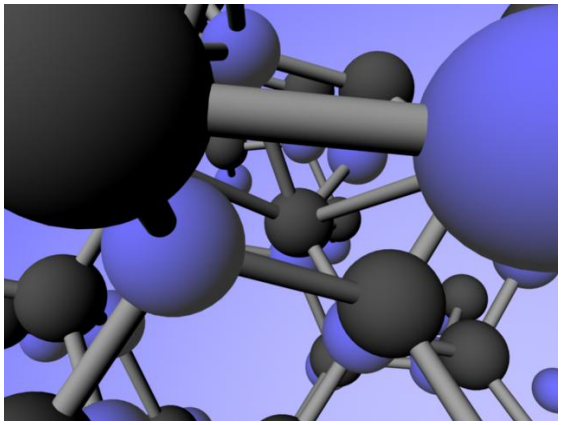

図 3.14: 異常部拡大図 1

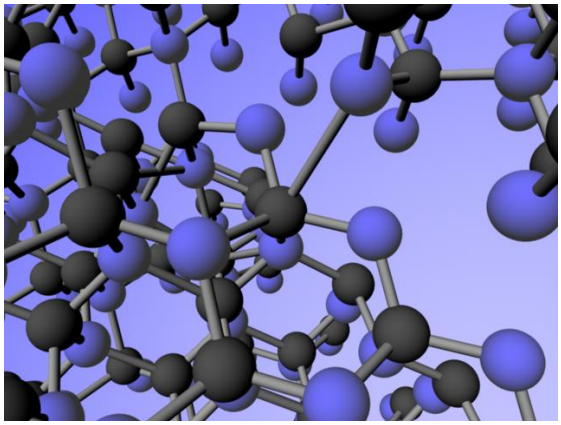

図 3.15: 異常部拡大図 2

3.3.4 四面体の特徴

他の四面体に対しても，抽出できていない異常がある可能性が出てきた．四面体１つ１つの特徴を調べるために，それぞれの四面体に対して以下の項目について調べた

四面体について

- 正常か異常か
- 正常判定の四面体の数
- 異常判定の四面体の数

四面体の中心について

- 高さ
- らせんの中心からの距離

四面体の頂点について

- 原子の種類
- 四面体中心からの距離
- 高さ
- らせんの中心からの距離

四面体の各面について

- 三角形の角度

らせんの中心から離れた四面体は特に問題なく構成されていた．特徴的だったのは，らせんの中心付近の正常な四面体である．中心付近の四面体で正しく四面体が構成されているもののほとんどが，四面体の中心と同じ原子を１つまたは２つ含んでいた．それらの中心からの距離は，他の頂点までの距離とほぼ等しかった．同じ原子がこれほど近づいてしまっているということは明らかに異常である．しかし，四面体の表示法では正常と判定され見落とされていた．異常判定された四面体は，距離が遠い頂点を持っていたが，その数はどれも１つで，２つ以上持っている四面体は存在しなかった．角度は，ばらつきはあったものの，正常と異常の間に大きな違いは見られなかった．よって中心付近の原子の結合はほとんどが乱れていることがわかった．

3.4 SiC結晶成長の機構

らせんの中心の結合の乱れは，半導体素子材料としてのSiC単結晶にとって大きな影響を及ぼす．特に，転位のバーガースベクトルが非常に大きくなった時，マイクロパイプと呼ばれる欠陥が現れ，致命的な欠陥となってしまう．現在，SiC単結晶の生成法として，液相成長法と気相成長法が提案されているが，結晶の成長機構はまだまだ不明な点が多い．らせん転位は欠陥ではあるが，マイクロパイプ欠陥や，中心付近の結合の乱れ，転位のすべり面が，結晶成長の機構に大きく関係していると考えた．

3.4.1 結晶成長

結晶が成長する時，結晶の表面に原子または分子が取り込まれて大きくなっていく．その表面がどのような状態にあるかが，結晶成長に影響を及ぼす重要な要素になる．表面には，テラス，ステップ，キンクと呼ばれる場所が存在する．表面の平な部分をテラスといい，高さの違うテラス間の階段状になっている所をステップという．また，ステップが折れ曲がっている所をキンクと呼ぶ．まず，結晶の表面に来た原子はテラス上を移動し，ステップに付着する．さらに，ステップにそって移動しキンクで安定する．つまり，キンクが多くあるほど結晶は成長しやすく，少ないほど成長しにくいと言える[4]．また．面自体のエネルギーも成長に影響を及ぼす．安定な面は成長しやすく，安定な面が広がっていくように成長する．

3.4.2 マイクロパイプ欠陥（ホローコア転位）

現在得られているSiC単結晶の最大の問題として，マイクロパイプと呼ばれる欠陥の存在がある．直径数mmの中空貫通欠陥であるマイクロパイプ欠陥は，エピタキシャル薄膜成長の際に引き継がれ，デバイス，特に大電力デバイスにとっては致命的な欠陥となる．この欠陥は，最近の研究によりFrankが1951年に理論的に予言したホローコア転位であることが明らかになってきた．ホローコア転位は，転位のバーガースベクトルが非常に大きくなったために転位芯が中空状になったものである．図3.16は原子間力顕微鏡で確認された，6H-SiC単結晶の成長表面に現れたマイクロパイプ欠陥に起因した渦巻成長模様である．渦巻成長中心に見える黒い孔がマイクロパイプ欠陥である．渦巻ステップの高さは13.5nmで，6H-SiCの格子定数の9倍にもなっている[5]．4H，6H-SiCのバーガースベクトルは基本的に大きく，マイクロパイプが形成され，原子レベルの非常に小さな穴が現れるのは珍しくない．しかし，図3.16のマイクロパイプは大きく，転位以外の要因が関わっているといえる．その要因こそ，結晶成長の機構ではないかと予測した．

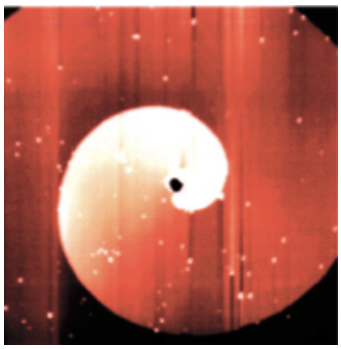

図 3.16: マイクロパイプ欠陥 [5].

3.4.3 らせん転位が存在する SiC の成長機構

図 3.17 は，西谷研究室の藤井が計算した，多くの C 原子が供給される気相成長法での SiC の表面エネルギーを示している．この図からわかることは，気相成長法では，(0001) 面は明らかに不安定であるということである．また (0001) 面に垂直な面である，(11-20) 面と (1-100) 面が．安定していると言える．

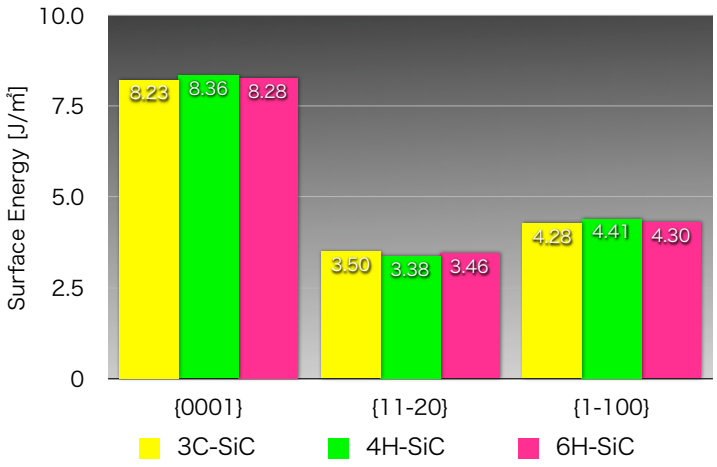

図 3.17: 気相成長中の Chemical Potential を用いて求めた表面エネルギー

図 3.18 はマイクロパイプが存在しているらせん転位モデルである．この図ではマイクロパイプの断面は (1-100) 面である．気相成長法では，(0001) 面より (0001) 面に垂直な面の方が安定していることがわかった．つまり，(0001) 面に対して垂直方向に結晶は成長していくことが考えられる．そうすると，中心のマイクロパイプ欠陥は残ったまま結晶が生成されてしまう．さらに，中心では原子結合が乱れていることから，その付近のエネルギーは高くなっていると予想できる．原子はエネルギーが低い，安定した場所で取り込まれようとするので，中心付近よりも中心から離れた場所に取り込まれる．その場所として挙げられるのが，らせん転位によってできるすべり面のステップである．すべり面も (0001) 面に垂直であるから，表面のエネルギーは安定しているといえる．このような結晶成長が続くと，中心付近の結合が乱れる範囲内では，原子は取り込まれないまま，穴の直径は広がりながら結晶が成長してしまう．その結果として，図 refmicro に示されているような大きなマイクロパイプの形成につながると予測することができる．よって，SiC 単結晶の成長機構として，気相成長法を用いると，マイクロパイプ欠陥が現れる可能性が高く，高品質な SiC 単結晶の生成は難しい．

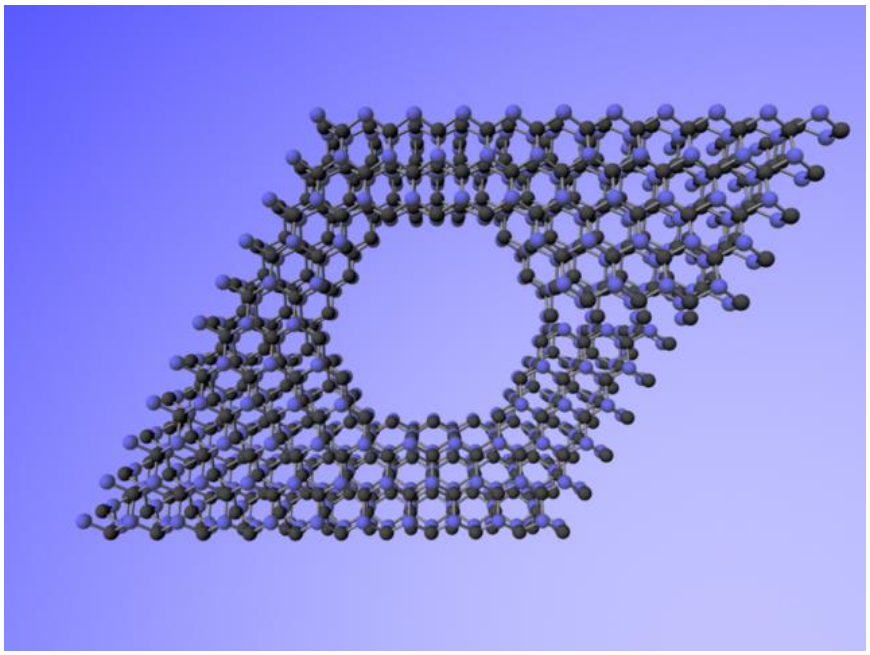

図 3.18: ホローコアモデル

第4章　総括

本研究によって得られた結果を以下に記す.

- Ruby スクリプトを使用し，原子位置等のデータを読み込み，MEL を出力させた．この方式を用いると，いくつかの利点がある．まず，同形式のデータならば，読み込ませるデータを変更するだけで表示することができる．そして，MEL スクリプトを大きく修正するとき，Ruby で MEL を出力させるようにしていると，Ruby スクリプトを少し変更するだけでよい．このように，MEL だけでは膨大な時間がかかる作業であっても，Ruby を用いると短時間でできるので非常に効率的である.

- 4H-SiC のらせん転位を視覚化することができた．様々なモデル作成を行い，Maya の自由な視点で図を見ることにより，らせん転位の中心付近の結合形態について知ることができた．また，視覚化されたモデルと，計算ソフトによる結果を合わせることにより，らせん転位が存在する SiC の成長機構について予測した．多くの C 原子が供給される気相成長法では，マイクロパイプの直径が広がり，欠陥が大きくなりつつ結晶が成長してしまう可能性があると予測することができた.

- 4H-SiC だけでなく，条件が合えば，本研究で作成した関数を使用することにより，様々な物質のらせん転位の視覚化や四面体表示を行うことができる.

謝辞

本研究を遂行するにあたり，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，西谷研究員の皆様にも様々な知識の供給，ご協力を頂き，本研究を大成する事ができました．この場を借りて心から深く御礼申し上げます．

関連図書

[1] 由宇義珍,『初めてのパワーデバイス』,(工業調査会，2006)．

[2] 阿部智弘,『MEL 教科書ー Maya プログラミング入門ー』,(株式会社ボーンデジタル，2004)．

[3] Chris Pine，西山伸,『初めてのプログラミング』，(株式会社オライリージャパン，2007).

[4] 小西孝典,『結晶成長プロセスの MAYA による視覚化』,(卒業論文，2008)．

[5] 大谷昇，藤本辰雄，勝野正和．矢代弘克,『大口径・高品質 SiC バルク単結晶開発の現状』,(新日本製鐵株式会社　先端技術研究所，2001)．