卒業論文

三面図を利用した粒界原子配列の視覚化

関西学院大学　理工学部　情報科学科

1549 成田 大樹

2017 年 3 月

指導教員　　西谷　滋人　教授

目次

目次

1　概要

本研究では，小傾角粒界エネルギーにおいて Hasson らによる Morse ポテンシャルを用いたシミュレーションの結果と大槻による実験結果の矛盾を原子レベルシミュレーションで解明することを目的としている．この矛盾を解明するためには，構造緩和をおこなって最安定の構造を求めることが重要になる．構造緩和が適切におこなわれるために，原子配列の視覚化を容易にするソフトが必要であり，そのソフトを開発していくことが本研究の目標である．

2　研究の目的

本研究の目的は，小傾角粒界エネルギーにおいて Hasson らによる Morse ポテンシャルを用いたシミュレーションの結果と大槻による実験結果の矛盾を原子レベルシミュレーションで解明することである．両者の具体的な相違点として，Hasson らによるシミュレーション結果では，0 度及び 90 度付近における対称傾角粒界エネルギーの立ち上がりがそれぞれ異なる傾きになった. その一方，大槻の実験結果では，0 度及び 90 度付近の対称傾角粒界エネルギーの傾きが左右対称になった.

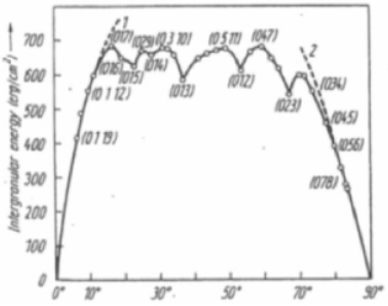

図 1　Hasson らによるシミュレーション結果.

{{attach_view(boundary_narita.003.jpg)}} 小傾角粒界エネルギーによる 2 つの結果の矛盾を明白にするために，西谷研究室において様々な検証が今までおこなわれてきた．はじめに，第一原理計算ソフト VASP による計算をおこない，その後，原子間ポテンシャルを使ったシミュレーションや Sutton Vitek による粒子モデルの研究を取り組んできた．また，八幡の研究では，経験的原子間ポテンシャルによるシミュレーションをおこない，Read-Shockley の理論予測と同様の結果となった [1]．さらに，岩佐の研究では，最安定な原子配置を探索するために原子の削除操作を取り入れ，第一原理計算ソフト VASP を用いて構造緩和し，系全体のエネルギー計算をおこなった．その結果，小傾角粒

界エネルギーが予測通りに大槻の結果よりも低いエネルギーの値となり，粒界エネルギーの立ち上がり 0 度付近が最安定の構造になった．しかしながら，粒界がより低い角度になった状態を計算していたため，図図:004 のように原子が全体的に傾いた構造になり，全ての構造が最安定であることの検証までには至らなかった．エネルギー計算による誤った構造緩和は，安定構造の原子配置を視覚的に確認しなかったことが原因である [2].

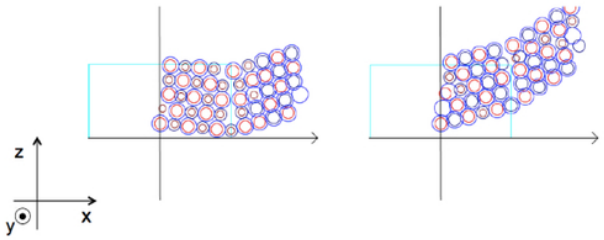

図 2　大槻による実験結果．岩佐による誤った構造緩和の結果

この失敗の原因を踏まえて，原子配置をより簡易的で視覚化できるプログラムを作成していく．したがって，粒界原子配列を正確で，且つ容易に視覚化できるためのソフトを開発することが本研究の目標である．本研究では小傾角粒界の原子モデルの配列を 2 次元で描画するソフトを開発する．

3　手法

本研究で開発するソフト”viewer”は，小傾角粒界の原子モデルの配列を 2 次元で描画する．viewer は，VASP の入出力で採用されている POSCAR 形式のファイルを入力とし，原子配列を SVG で出力する．なお，ソフト開発は作業の分業化が容易である MVC モデルで作成していく．

3.1　MVC モデルの概要と利点

ソフトウェアの設計モデルの一つであり，アプリケーションの開発において取られている手法である．MVC モデルは，データの処理を担う”Model”，処理結果を画面に表示する”View”，入力情報の受け取って処理機能を制御する”Controller”の三要素で構成されている．MVC モデルで開発することにより，各機能が直交化され，開発作業を分業化しやすく，相互の仕様変更による影響を受けずに開発を進めることが出来る [3]．したがって，原子配列の結果を画面表示する機能構築に特化した開発が可能となる．

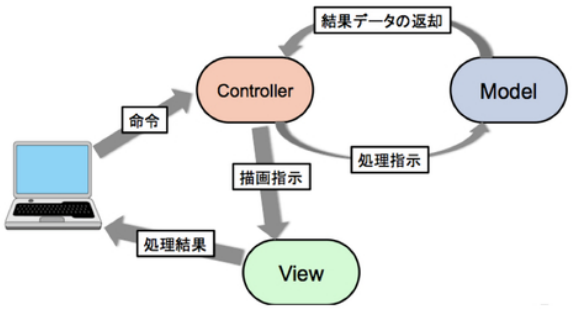

図 3　MVS モデルによる viewer の位置.

3.2 SVG の特徴

SVG は，XML を基盤とする 2 次元画像記述言語であり．イラストレーターで扱うベクターデータである [4]．SVG で表示することで，以下のような利点がある．

- ベクタ形式で画像を表示するため，曲線や文字の拡大・縮小しても画質が劣化せずに，解像度に応じた出力結果を得ることが出来る
- スタイルシートを切り替えることで，特殊な環境下においても目的に応じたグラフィック描画を出力することが可能である
- テキストデータであるため，テキストエディタや xml プロセッサを介したファイルの編集が可能である

なお，原子配列の SVG 表示は，2 次元画像描画ライブラリ”rcairo”を用いて作成していく．

3.3 rcairo の使用

rcairo は，Ruby 言語でベクタ形式の 2 次元描画を容易に実現できるライブラリのことである．このライブラリは，描画コンテキストを用いた API であるため，描画するためのコードを短く簡潔に作成することが出来る．また．複数の出力をサポートすることができるため，出力先のフォーマットに影響されずに描画処理をおこなうことが可能である [5]．

3.4 一般的な原子モデルソフトとの比較

結晶描画に特化したソフトとして，”Medea”や”VESTA”がある．結晶構造や電子・核密度等のデータを読み込んで結晶の形を三次元で可視化できるプログラムである [6]．これらは原子の配置を確認するために，手作業で結晶構造の視点を自由に変えながら．構造全体を視覚的に理解することができる．図には VESTA の画面から切り出した原子配置モデルの一例を示している．3 次元に投影することによって得られた図形である．VESTA の使用には以下の難点があった．

- 三次元で原子配列を表示するため，各層の原子位置が把握しづらい
- 色分けするための層を手動でおこなうため，指定した範囲の正確さに欠け，作業に手間がかかる

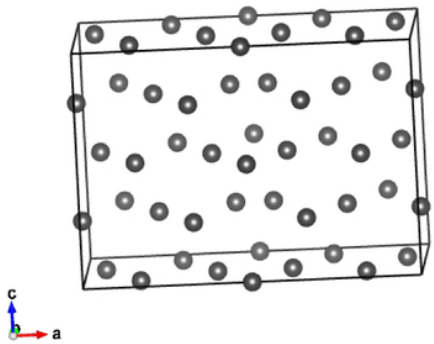

図 4　VESTA で描画した POSCAR_2223 の原子配置.

4　結果

4.1 結晶構造描画ソフト"VESTA"

VESTA は，結晶構造や電子・核密度等のデータを読み込んで結晶の形を三次元で可視化できるプログラムである [6]．手作業で結晶構造の視点を自由に変えながら．構造全体を視覚的に理解することが出来るが，VESTA の使用には以下の難点があった．

- 三次元で原子配列を表示するため，各層の原子位置が把握しづらい
- 色分けするための層を手動でおこなうため，指定した範囲の正確さに欠け，作業に手間がかかる

図:006

4.2 三面図による描画

三面図は，立体を正面図，平面図，側面図の三方向からみて投影した図を展開したもので，立体の形状を 2 次元上で適切に表示することが出来る．三方向から描く各図の位置は厳密に決められており，正面図を物体の最も代表的な面とし，正面図の真上に平面図，正面図の真横に側面図を描く [7]．実際に，三面図を使用して表示した POSCAR_2223 の成功例と失敗例は図のようになる．

図:008 図:009

これまでの原子配列の構造は，結晶構造描画ソフト VESTA を使用して確認してきたが．三面図を使用して表示することにより，図図:007 のように，各面から原子の配置を正確で簡易的に確認できるようになった．

図:007

4.3 viewer を構成する関数と各々の機能

4.3.1 外部データの読み込み部

■read_pos

```
def read_pos(lines, init_line=8)
  lattice, atom, poscar = [],[],[]
  lines[2..4].each{|line| lattice << line.scanf("%f %f %f\n")  }
```

```
  lines[init_line..lines.length+1].each{|line| atom << line.scanf("%f %f %f\n"

  atom.each{|i_atom|
    pos=[0.0,0.0,0.0,0.0]
    i_atom.each_with_index{|atom_j,j|
      lx,ly,lz=lattice[j]
      pos[0] += atom_j*lx
      pos[1] += atom_j*ly
      pos[2] += atom_j*lz
    }
    poscar << pos
  }
  return poscar
end
```

外部データである POSCAR_2223 を ARGV で読み込み，lines として定義している．lines の 2 行目から 4 行目の値を読み込んで，配列 latttice に格納している．lines の 8 行目から最後の行の値を読み込んで，配列 atoms に格納している．配列 lattice 内に格納された各々の座標を lx,lylz と定義し，格子の座標と原子の座標を掛け算して粒界配列の座標を出し，その値を poscar に格納している．{{br}} poscar に格納された原子座標が以下の実行結果である．

```
[Narita-no-MacBook-Air:~/boundary/narita/view] nari% ruby final.rb POSCAR_2223
[5.75101295815, 0.0, 0.0, 0.0]
[7.668017277916733, 0.6390014395849836, 2.020699977875, 0.0]
[3.834008638383265, 0.6390014395849836, 2.020699977875, 0.0]
[7.029015837611088, 2.556005758968566, 0.0, 0.0]
[4.473010078688911, 2.556005758968566, 0.0, 0.0]
[5.75101295815, 0.0, 4.04139995575, 0.0]
[7.668017277916733, 0.6390014395849836, 6.062099933625, 0.0]
[3.834008638383265, 0.6390014395849836, 6.062099933625, 0.0]
[7.029015837611088, 2.556005758968566, 4.04139995575, 0.0]
[4.473010078688911, 2.556005758968566, 4.04139995575, 0.0]
[6.390014398455645, 4.473010078352149, 2.020699977875, 0.0]
```

```
[5.112011517844354, 4.473010078352149, 2.020699977875, 0.0]
[6.390014398455645, 4.473010078352149, 6.062099933625, 0.0]
[5.112011517844354, 4.473010078352149, 6.062099933625, 0.0]
...
[1.2780028794610885, 5.751012958150748, 6.062099933625, 0.0]
```

4.3.2 原子座標の計算部

■identical_atoms

```
def identical_atom(i_atom,j_atom)
  dist=0.0
  3.times{|i| dist += (i_atom[i]-j_atom[i])**2  }
  return true if Math.sqrt(dist)<0.5
  return false
end
```

関数 identical_atoms は，２つの POSCAR ファイルに含まれる各々の原子座標の距離を求めて，ある値の大小比較をおこなって真偽を判定する機能を果たしている．具体的には，ファイル内のある原子 i_atom ともう片方のファイル内にある原子 j_atom の距離が 0.5 より小さい値であれば true と判定して返す．そうでなければ，false を返して終了する．

■mk_deleted_atom

```
def mk_deleted_atom
  mark=[]
  j_max = $pos_after.length
  $pos_before.each_with_index{|i_atom,i|
    update_num=0
    $pos_after.each_with_index{|j_atom,j|
      break if identical_atom(i_atom,j_atom)
      update_num = j
    }
    mark << $pos_before[i] if update_num==(j_max-1)
```

```
    }
    return mark
  end
```

関数 mk_deleted_atom は，原子削除をおこなう前後の POSCAR ファイルを比較して削除された原子のみを別の配列に格納する機能を果たしている．関数内では，始めに更新するための値 update_num を 0 と定め，２つの POSCAR ファイルを each 文で繰り返しながら，identical_atom で true として返されたものを配列 mark の方へ格納している．{{br}} 原子座標ファイル POSCAR_2223 において，配列 mark に格納された原子座標が以下の実行結果である．

```
[Narita-no-MacBook-Air:~/boundary/narita/view] nari% ruby final.rb POSCAR_2223
[[6.390014398455645, 4.473010078352149, 2.020699977875, 0.0]
[5.112011517844354, 4.473010078352149, 6.062099933625, 0.0]
[10.863024475994353, 3.8340086387671652, 0.0, 0.0]
 [0.6390014403056455, 3.8340086387671652, 0.0, 0.0]
[10.863024475994353, 3.8340086387671652, 4.04139995575, 0.0]]
```

4.3.3 粒界原子の描画部

■draw_backcolor, draw_exes

```
  def draw_backcolor
    $context.set_source_rgb(0.8, 0.8, 0.8)
    $context.rectangle(0, 0, $width, $height)
    $context.fill
  end

  def draw_axes
    $context.set_source_rgb(0, 0, 0)
    [[0,1],[1,0]].each{|line|
      x,y=line[0],line[1]
      [[0,0],[$cx,0],[0,$cy]].each{|c_x,c_y|
        $context.move_to($mv+c_x,$mv+c_y)
        $context.line_to($mv+c_x+x*$scale,$mv+c_y+y*$scale)
```

```
      $context.stroke
    }
  }
end
```

関数 draw_backcolor は，原子配列の背景を描画しており，SVG で表示する画面のサイズを明確にするために作成している．また，関数 draw_axes は，原子配列における縦軸と横軸を描画しており，each 文で 3 つ平面に必要な軸をまとめて出力できるようにしている．

■open_circle

■draw_each_plane, pos_y

```
def draw_each_plane(ind_1,ind_2,c_x,c_y)
  rr = 2
  layer_num = 3
  sel = (ind_1==0 and ind_2==1)? 1 : 0
    [[$deleted_atoms,[1,0,0],rr*1.3],[$pos_after,[0,0,1],rr]].each{|atoms_colo
    $context.set_source_rgb(atoms_color[1])
    radius = atoms_color[2]
    atoms_color[0].each{|pos|
      if open_circle[layer_num-1] < pos[2] && pos[2] < open_circle[layer_num]
        $context.circle($mv+c_x+$adjust*pos[ind_1],pos_y(pos,c_y,ind_2,sel), r
        $context.stroke
        $context.set_line_width(0.5)
      else
        $context.circle($mv+c_x+$adjust*pos[ind_1],pos_y(pos,c_y,ind_2,sel), r
        $context.fill
      end
    }
  }

  if $pos_before.size==$pos_after.size
    $context.set_source_rgb(0, 0.8, 0)
```

```
    (0..$pos_before.length-1).each{|i|
      $context.move_to($mv+c_x+$adjust*$pos_before[i][ind_1],pos_y($pos_before
      $context.line_to($mv+c_x+$adjust*$pos_after[i][ind_1],pos_y($pos_after[
      $context.stroke
    }
  end
end

def pos_y(pos, c_y, index, select)
  dy = select == 0 ? pos[index] : $pos_max[index]-pos[index]
  return $mv+c_y+$adjust*dy
end
```

関数 draw_each_plane で，描画する原子の位置，色，大きさを定めており，関数 pos_y は x − y 平面での y 軸を逆転する計算をおこなっている．具体的に，sel で描画する平面の判定をおこない，x − y 平面ならば，関数 pos_y で一番大きい y 座標の値から各 y 座標を差分した値を受け取っている．layer_num は，白抜きする z 層を指定しており，関数 open_circle をもとに白抜きする原子を判定している．また，読み込んだ２つの原子配列ファイル POSCAR のデータサイズが同じならば，各原子の位置に相当する原子との距離を線で表示する．

■draw_atoms

```
def draw_atoms
  draw_each_plane(0,1,0,0)
  draw_each_plane(0,2,0,$cy)
  draw_each_plane(1,2,$cx,$cy)
end
```

関数 draw_atoms では，前の関数 draw_each_plane で設定した原子を三面図の規定位置にそれぞれ描画するようにしている．具体的には，x − y 平面を平面図，x − z 平面を正面図，y − z 平面を側面図として三面図を構成させている．

■find_max

```
def find_max(pos)
```

```
  max = [0,0,0]
  [0,1,2].each{|ind|
    pos.length.times {|i| max[ind] = pos[i][ind] if max[ind] < pos[i][ind] }
  }
  return max
end
```

関数 find_max は，原子の各座標の最大値を探索する機能を果たしている．配列 pos には，関数 read_pos で (x,y,z)=(0,1,2) としてそれぞれの値を読み込んでおり，隣接する値の大小比較をおこなって，各座標の最大値を配列 max に格納している．

■main_draw

```
def main_draw(file1,file2,   model_scale = 10)
  lines1 = File.readlines(file1)
  lines2 = File.readlines(file2)
  $pos_before = read_pos(lines1,8)
  $pos_after = read_pos(lines2,8)
  $deleted_atoms = mk_deleted_atom

   $pos_max=find_max($pos_before)
   $pos_max[0].ceil*10
  $width,$height = 300,200
  $cx,$cy = $width/2.0,$height/2.0
  $mv = 10
  $scale = 1000
  $adjust = $scale/($pos_max[0].ceil*model_scale)
  surface = Cairo::SVGSurface.new('view.svg', $width, $height)
  $context = Cairo::Context.new(surface)
  $context.set_line_width($line_width)

  draw_backcolor
  draw_axes
  draw_atoms
```

```
    surface.finish
  end
```

関数 main_draw は，読み込んだ外部ファイルを lines として格納し，関数 read_pos でつくった原子座標 poscar をファイルごとに分類している．また，rcairo で描画処理の基本となるサーフェスとコンテキストを作成し，出力場所及び出力先の形式を定めている．

5　総括

粒界原子配列を簡易的に視覚化するソフト”viewer”を作成することで，原子配列を 2 次元で，且つ 3 方向からの視点で SVG 表示した．その結果，構造緩和をおこなうための POSCAR ファイルの誤りを発見することが出来た．

6 参考文献

[1] 小傾角粒界粒子シミュレーションの原子ポテンシャル依存性, 八幡裕也 (関西学院大学 理工学部研究科情報科学専 攻 修士論文 2015)

[2] 原子削除操作を加えた対称傾角粒界のエネルギー計算, 岩佐恭佑 (関西学院大学 理工学部研究科情報科 学士論文 2016)

[3] MVC(Model-View-Controller) を理解する, CakePHP,https://book.cakephp.org/2.0/ja/cak

[4] SVG 入門, 新山祐介, コンピュータサイエンス入門 by 新山祐介, https://euske.github.io/eusk

[5] cairo:2 次元画像描画ライブラリ, 須藤功平, Rubyist Magazine-るびま Vol.54 (2016-08), http://magazine.rubyist.net/?0019-cairo

[6] VESTA, Koichi Momma(2004-2017), JP-Minerals, http://jp-minerals.org/vesta/jp/

[7] 三面図 (機械設計のための基礎製図), 独立行政法人 海上技術安全研究所, NMRI, https://www.nmri.go.jp/eng/khirata/mechdesign/ch04/ch04.html