卒業論文

粒界モデル表示のTkinter版の作成

関西学院大学理工学部

情報科学科 西谷研究室

学籍番号 上山竜司

2023年3月

概 要

西谷研では，粒界に存在する転位の性質を解明するために第一原理計算を実行してきた．合金の脆さの原因は粒界偏析と考えられている．偏析は粒内，粒界のどちらでも起こるとされているが，傾向として粒界で起こりやすいと考えられる．固溶エネルギーは化学結合と原子半径で決定されるため，熱膨張が起こるとエネルギーが変化する．

しかし，系全体のエネルギー変化・粒界の位置関係だけでは，転位あるいは粒界近傍でどのような変化が起こっているのかを検討することが難しかった [1]．このため，モデルの原子位置や計算結果のエネルギーのカラー表示などの視覚化ツールが開発されてきた．

いままでは，傾角粒界 (tilt) のエネルギー差のスペクトル表示 [2] やねじれ粒界 (twist) を見やすく表示 [3] するために描画機能に優れた Processing 言語を用いていた．しかし，より多くの人に利用してもらうためには，普及した言語で書かれている方が需要が高いと考えた．そのため，本研究では Python でリライトし，粒界モデルを表示するアプリケーションを作成した．

目 次

図 目 次

第1章　序論

西谷研では第一原理計算を実行してきた．これは，粒界に存在する転位の性質を解明するための計算である．合金の脆さの原因は，粒界偏析と考えられている．偏析とは，溶質が偏って分離し，その周りに析出物ができてしまうことである．この現象は粒内，粒界のどちらでも起こるとされているが，傾向として粒界で起こりやすいと考えられている．固溶エネルギーは化学結合と原子半径で決定されるため，熱膨張が起こるとエネルギーが変化する．粒界では該当するサイトに対する隣接原子との距離から考えると，周りの原子との距離が遠いところでは膨張が起こらずエネルギーが低下し，距離が近いところでは膨張し，エネルギーが上昇すると考えられる．

しかし，系全体のエネルギー変化・粒界の位置関係だけでは，転位あるいは粒界近傍でどのような変化が起こっているのかを検討することが難しかった [1]． このため，モデルの原子位置や計算結果のエネルギーのカラー表示などの視覚化ツールが開発されてきた．

そこで，傾角粒界 (tilt) のエネルギー差のスペクトル表示 [2] やねじり粒界 (twist) を見やすく表示 [3] するために，描画機能に優れた Processing 言語を用いていた．しかし，Processing 言語はあまり普及していない．より多くの利用者に供するためには，より普及している言語で書かれている必要がある．

そのため，より需要の高い Python でのリライトを行い，粒界モデルを表示するアプリケーションの作成を本研究の目的とする．

第2章　手法

2.1　Pythonでの開発

本研究は，Processing言語で書かれたプログラムをPython版としてリライトしたものである．Processing言語はグラフや図形をアニメーションで実行できる点や機能が豊富にあるというメリットがあった．基本の描画機能のほか，デフォルトであらゆる機能が搭載されているため，ある程度のソースコードを書けばGUIが完成する．しかし，Processing言語は利用者が少ないこと，機能があらかじめ用意されているため自ら考えてコードを書かずに済んでしまうことなどがデメリットであると感じた．そのため，Processing言語以外でグラフィックなプログラムを作成できるPythonでの開発を行った．

2.2　Tkinter

Tkinterとは，Window，Mac OSX，Linuxといった主要なOSにも対応しているPythonに標準で付属しているGUIライブラリである．TkinterはTcl/Tkをベースにしたものである．

Tcl(Tool Command language)とは1988年にスプリクト言語として登場した．その後，TclでGUI開発を行うためのツールキットとしてTk(Tool Kit)が開発された[4].

2.2.1　開発環境の構築

開発環境としてOSXを想定したが，Tkinterを使おうとすると,「ModuleNotFoundError: No module named ‘ _tkinter ’」というエラーメッセージが表示される場合がある．

```
python3 BoundaryViewer.py
Traceback (most recent call last):
```

```
  File "/Users/bob/git_hub/boundary_viewer/BoundaryViewer.py", line 1, in <module>
    import tkinter as tk
  File "/opt/homebrew/Cellar/python@3.10/3.10.9/Frameworks/Python.framework/Versions/3
    import _tkinter # If this fails your Python may not be configured for Tk
ModuleNotFoundError: No module named '_tkinter'
```

上記は，環境が整っていない状態で Tkinter を import したプログラムを実行したときのエラー文となっている．この場合の解決法を以下に示す．

Homebrew を使って，python-tk をインストールする．

```
brew install python-tk
```

Python のバージョンに合わせたい場合は以下のようにバージョンを指定することもできる．

```
brew cleanup python-tk@3.10
```

2.3 第一原理計算

第一原理計算とは，原子位置を入力として電子構造をシュレディンガー方程式に従って計算し，系のエネルギーを出力する方法である．これにより，様々な物性を予測することが可能である．

2.3.1 VASP

VASP は第一原理量子力学計算を行うための計算化学ソフトウェアで，PAW 型擬ポテンシャル法と平面波基底を用いた第一原理計算パッケージである [1].

2.3.2 POSCAR

POSCAR ファイルは，第一原理計算ソフトである VASP で利用されており，格子定数や原子数，および各原子の相対座標を入力して，原子モデルを構築するために使用する

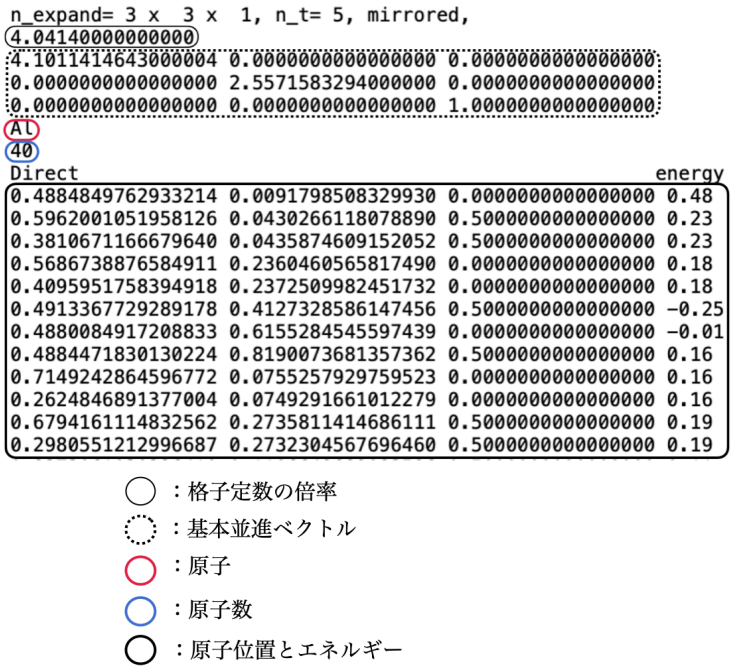

図 2.1: POSCAR ファイルの例.

ファイルである．このフォーマットは VASP に限らず，格子モデルの構築に広く利用されている．図 2.1に，POSCAR ファイルの例を示した．

実線で囲まれた部分は，格子定数の倍率である．倍率を変更することで，基本並進ベクトルも比例し，格子を膨張させることができる．点線で囲まれた部分は基本並進ベクトルで，1，2，3 行目は a，b，c のベクトルを表す．赤の太線で囲まれた部分は原子を表しており，図 2.1では Al（アルミニウム）となっている．青の太線で囲まれた部分は原子数を表し，黒の太線で囲まれた部分はそれぞれの原子の相対座標（x，y，z）とエネルギー値となっている．

2.3.3 周期的境界条件

結晶モデルのエネルギーの計算には結晶格子が周期的に並べられている理想的な状態であると仮定して行われる．有限的に広がっている結晶を表面がなく，計算で扱いやすい無限結晶として扱うことができるため，第一原理計算では周期的境界条件が必須である．本研究の場合は，POSCAR に書かれた原子数 40 が 1 セルに該当する．

第3章　結果と考察

3.1　アプリケーション使用法 - BoundaryViewer.py

3.1.1　構成

アプリケーション BoundaryViewer は図 3.1に概略を示したとおり，主に 3 つのプログラムからなっている．1 つは BoundaryViewer.py であり，このプログラムはアプリケーション画面のレイアウトや他の 2 つのプログラムを起動させるボタンの設定が書かれている．2 つ目，3 つ目は以降詳しく紹介する SiteEnergy.py と twist07.py である．

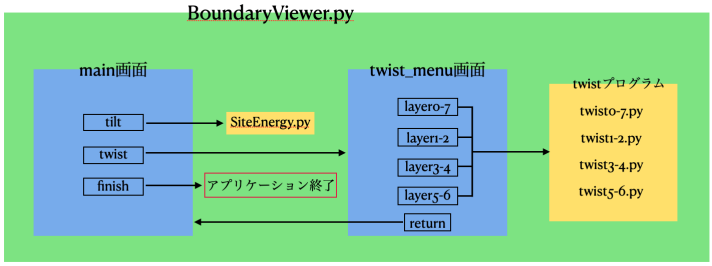

図 3.1: BoundaryViewer の概略図.

3.1.2　使い方

図 3.2はアプリケーションのメイン画面と画面遷移後である．tilt ボタンをクリックすると SiteEnergy.py のプログラムが起動し，図 3.7が表示される．finish ボタンをクリックとアプリが終了する．twist ボタンを押すと図 3.2のようにねじり粒界のメニュー画面に遷移する [9]．ねじり粒界のメニュー画面では表示したい layer を選び，それぞれのボタンを押すと図 3.3に対応したプログラムが起動するようになっている．

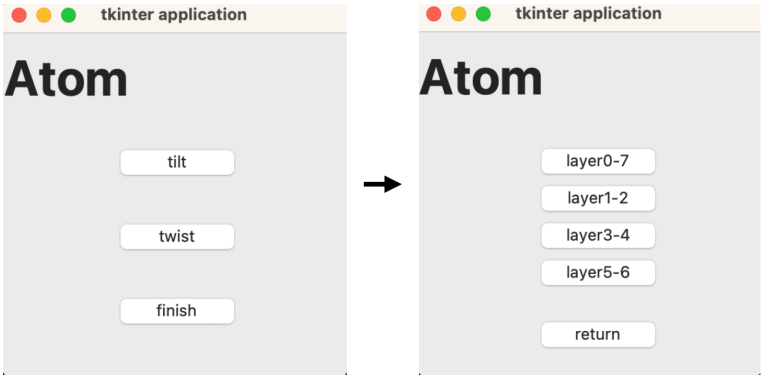

図 3.2: アプリケーション画面.

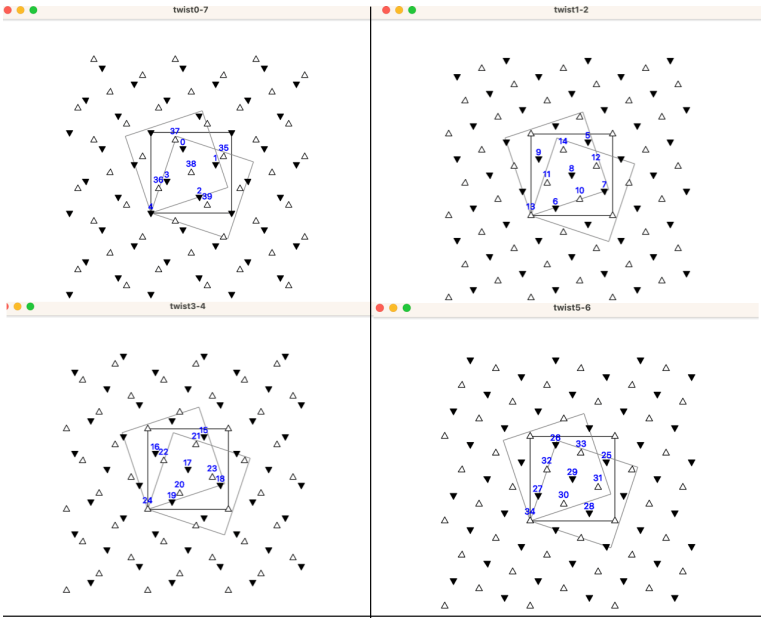

図 3.3: 各層のねじり粒界.

3.2 POSCARの読み込みと拡張表示

POSCARのデータをもとに粒界モデルの表示を行う．POSCAR読み込むコードは次のとおりである．

```
canvas = tk.Canvas(root, width = 600, height = 480, bg = 'white')
canvas.place(x = 0, y = 0)
root.update_idletasks()
xp = canvas.winfo_width()/2
yp = canvas.winfo_height()/2

with open('POSCAR_ene.txt', 'r') as f:
    lines = f.read().split("\n")
    title = lines[0]
    m = re.findall('\d+',title)
    n_t = int(m[3])
    nx = int(m[1])

    for i in range(0,3):
m = lines[i+2].split(' ')
la = float(m[i])
if i==0:
    x_lat = la * scale
if i==1:
    y_lat = la * scale

    m = re.match('[0-9]+',lines[6])
    n_atom = m[0]
    theta = atan(1.0/float(n_t))

    x0 = -0.5 * x_lat + xp
```

```
    y0 = 0.5 * y_lat + yp
    f.close()
```

canvas には画面サイズ (600x480) と色 (white) を指定している．画面サイズは描画するに場合によく使用するため，xp・yp にはサイズの半分の値をセットする．

まずは，POSCAR_ene.txt を開き，改行ごとに lines 配列にデータを入れている．次に lines を指定し図 2.1の点線部のベクトルや黒の太線部の原子位置など後に必要となるデータを変数に格納している．

正規表現については，qiita の記事を参考にした [10].

3.2.1 周期的境界条件の実装コード

周期的境界条件のコードを以下に記す.

```
def shift(x,y):
    vv = [0,0]
    vv[0] = x+x0
    vv[1] = -y+y0
    return vv[0],vv[1]


def matrix(theta, x, y):
    vv = [0,0]
    vv[0] = np.cos(theta) * x - np.sin(theta) * y
    vv[1] = np.sin(theta) * x + np.cos(theta) * y
    return vv[0],vv[1]


def multiple(theta, x, y, i, j, k):
    v0 = matrix(theta, (j+float(x))*x_lat, (k+float(y))*y_lat)
    v1 = shift(v0[0], v0[1])
    return v1
```

```
#main
for j in range(-1,2):
    for k in range(-1,2):
v1 = multiple(0.0,m[0],m[1],i,j,k)
atom = circle(v1[0],v1[1],rr)
atom.create(canvas)
```

3 つの shift, matrix, multiple 関数により，周期的に粒界を並べている．main 部分で multiple 関数に引数を入れると，配列 v0 に matrix 関数の戻り値，配列 v1 に shift 関数の戻り値が与えられる．multiple 関数の戻り値 v1 は x,y 座標であり，その位置に circle を描画する．

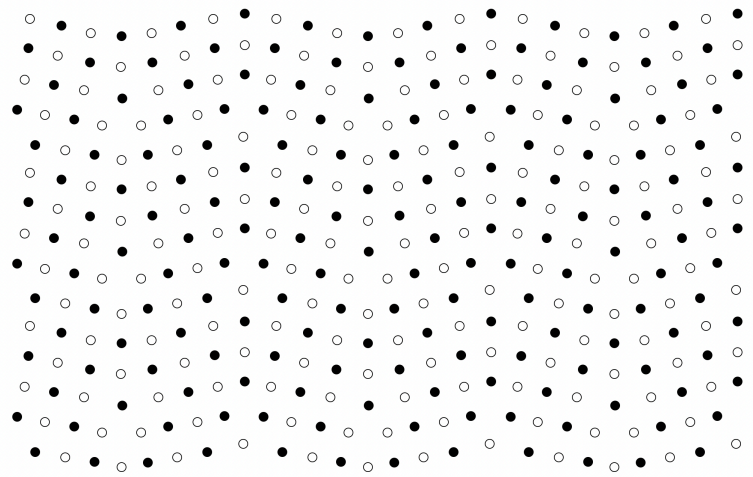

図 3.4: 周期的境界条件での拡張.

POSCAR の原子数のデータ 40 よりも多く表示されているのが分かる．次の節では図 3.4にエネルギーを加味し，スペクトルで色をつける．

3.3 傾角粒界 (tilt) のスペクトル表示 - SiteEnergy.py

図 3.4やエネルギー変化の数値だけでは，転位あるいは粒界近傍でどのような変化が起こっているのかを検討することが難しい．つまり，数値だけでは直感的に理解することは困難である．

そのため，粒界のエネルギー差をスペクトルで表示し，各サイトでの安定・不安定な場所を視覚化する．

3.3.1 HSV から RGB への変換

Tkinter での色 (color) の設定方法は 2 つある．

1 つは，表 3.1で示した RGB の指定による方法である．RGB(赤，緑，青) はそれぞれ 16 進数で表される．そのため，表 3.1の#rgb 形式では個々の色が 16 色，つまり 2^4(=16) で表すことができることから 4 ビットカラーと呼ばれる．同じように#rrggbb は 16x16 の 256(=2^8) 色ずつで表すので 8 ビットカラー，#rrrgggbbb は 16x16x16 の 4096(=2^{16}) 色なので 16 ビットカラーという．

表 3.1: RGB の指定による方法.

#rgb	4 ビットカラー 16 色
#rrggbb	8 ビットカラー 256 色
#rrrgggbbb	16 ビットカラー 4096 色

2 つ目は，色名称による設定方法である．全ての環境において使用可能な色名称

'white', 'black', 'red', 'green', 'blue', 'cyan', 'yellow', 'magenta'

などを書いて設定する [5].

本研究では，粒界エネルギー最小値・最大値を使い，その値をスペクトル表示に使用するため，RGB の指定による色の設定方法を活用する．

説明したとおり，Python には HSV による色の設定方法がないため，HSV から RGB へ変換するプログラムを作成する．

1．Python による color 実装コードの解説

```
def RGB(h):
    R = 0
    G = 0
    B = 0
    max = v
```

```
    min = max - ((s / 255) * max)
    if h < 60:
R = max
B = min
G = (h / 60) * (R - B) + B

    elif h >= 60 and h < 120:
G = max
B = min
R = ((120 - h) / 60) * (G - B) + B

    elif h >= 120 and h < 180:
G = max
R = min
B = ((h - 120) / 60) * (G - R) + R

    elif h >= 180 and h < 240:
B = max
R = min
G = ((240 - h) / 60) * (B - R) + R

    elif h >= 240 and h < 300:
B = max
G = min
R = ((h - 240) / 60) * (B - G) + G

    elif h >= 300 and h < 360:
R = max
G = min
```

```
B = ((360 - h) / 60) * (R - G) + G

return R,G,B

def set_color(R,G,B):
    return "#" + hex(int(R)).lstrip("0x").zfill(2) +
   hex(int(G)).lstrip("0x").zfill(2) +
   hex(int(B)).lstrip("0x").zfill(2)
```

この関数ではHSV(色相，彩度，明度)をRGBに変換するものである．本研究では8ビットカラーを使用するため，彩度を表す変数 s (Saturation)と明度を表す変数 v (Value)は最大値255としている．色相を表す引数 h (Hue)による条件のもと，R,G,Bを返す[6].

set_color関数はRGB関数の戻り値R,G,Bを16進数に変換する．Rが255のとき，hex(R)とすると0xffと変換できる．#rrggbb形式にするためには0xをとり，2桁ずつの表示にする必要があるので，

```
hex(int(R)).lstrip("0x").zfill(2)
```

となる．同様に引数G,Bも16進数に変換することで，#ff0005のような8ビットカラーが得られる[7].

2．色相環(0-360度)

```
Range = 360
for i in range(0,Range):
    color = RGB(i)
    canvas.create_arc(20,20,400,400,
    start=90-i,extent=-1,fill=color,outline=color)
```

このコードを加えると図3.5のようにスペクトルが0°の赤(#ff0000)から始まり，緑(#00ff00)，青(#0000ff)を通り，360°の赤に戻る色相環が構成される．

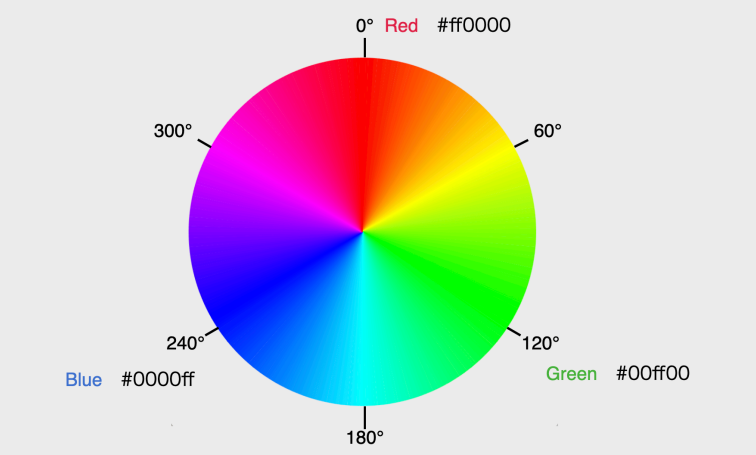

図 3.5: スペクトル表示.

3．色相環 (0-240 度)

色彩の温度感は身近なものとして「暖色」と「寒色」がある．本研究の目的は粒界の様子を視覚化することであるため，赤 (#ff0000) から緑を通り，青 (#0000ff) までの領域だけを使って表示することとする．

そのため，色相環の 0 度から 240 度でのスペクトル変化を取り出す．変数 Range を 240 に変更する．

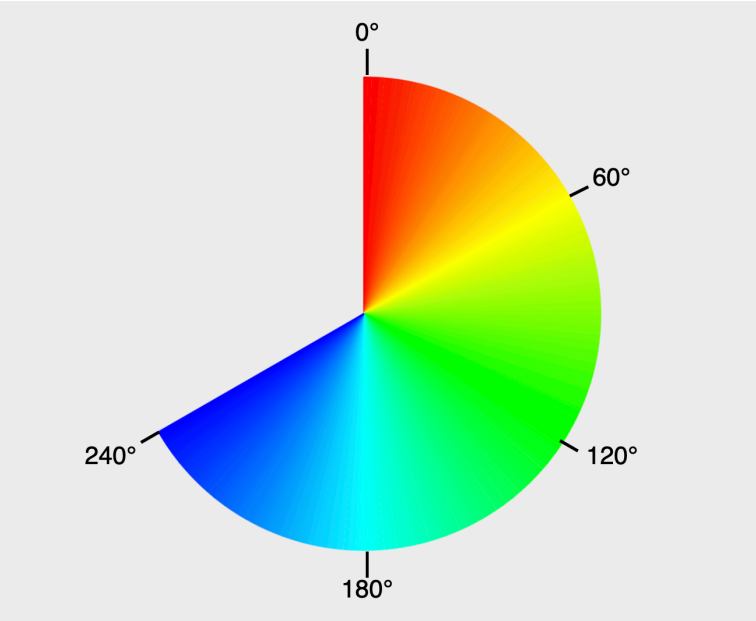

図 3.6: 0 度から 240 度までのスペクトル表示.

0 度から 240 度までのスペクトルを使い，POSCAR ファイル (図 2.1) の energy から，

最小値・最大値をスペクトル表示の範囲に変更する．そして，エネルギー値ごとの割合により色を変化させる．

割合を求めるコードは以下の通りである．

```
def ratio():
    min = 10000
    max = -10000
    for i in range(0,int(n_atom)):
m = lines[i+8].split(' ')
if (min > float(m[3])):
    min = float(m[3])
if (max < float(m[3])):
    max = float(m[3])


    dif1 = (max - min) * 100
    ratio = 240 / dif1
    return ratio, max


Ratio,MAX = ratio()
dif2 = (atom_energy - MAX) * 100
hsv = (abs)(dif2 * Ratio)
```

4．ratio 関数

n_atom には POSCAR の原子数 40 が入っており，for 文を 40 回まわす．energy を 40 個調べ，最小値 min と最大値 max を決定する．スペクトルの範囲をエネルギー値に変更するため，最小値と最大値の差 (dif1) を取る．先ほどスペクトル表示は 240 度まで使用するとしたので，240 を dif1 割ることで最大値からの差の割合が求まる．これを ratio とした．

5．main

atom_energy は POSCAR で読み込んだ原子それぞれのエネルギーであり，エネルギーの最大値との差（dif2）に Ratio を掛けると 240 までの値が得られる．これを hsv としている．

3.3.2 完成図

前章の図 3.4に色をつける．

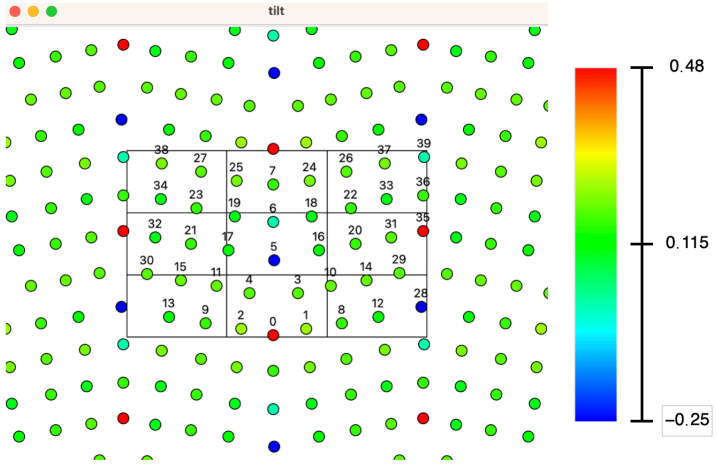

図 3.7: 傾角粒界 (tilt) のスペクトル表示.

2 層 500K の時の Al を Mg に置換した図である．サイト 0 は+0.48eV 上昇し，サイト 5 は-0.25eV エネルギーが低下したことがわかる．Mg の原子半径は Al よりも大きく，粒界転移の中心となるサイト 5 は空隙が大きく，looser なサイトであることから青色で表示されており，安定していることがわかる．赤色となっているサイト 0 は隙間が狭い tighter なサイトで，エネルギーが上昇していることから不安定となっている．

3.4 ねじり粒界 (twist) の表示 - twist07.py

図 3.8はねじり粒界の 3x3 モデルを示したものである．ピンクの面で示した部分が粒界（Grain Boundary）である．本研究では，ねじり粒界を 2 層ごとの平面図として描画することとした．そして，各層のサイトナンバーを描画することで等価なサイトを視覚的に判断することが可能となる．

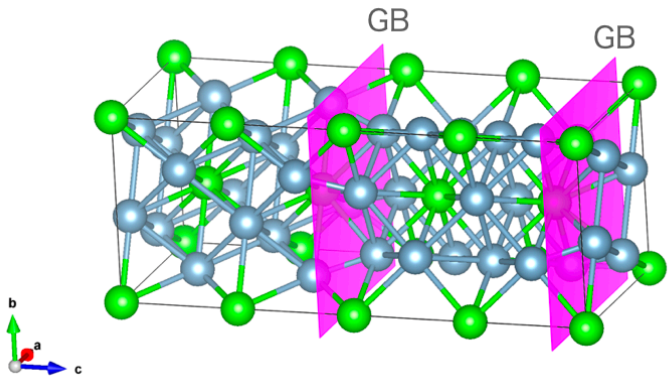

図 3.8: ねじり粒界の 3x3 モデル.

Tkinter には三角形を描画する関数が存在しないため，ねじり粒界の 3x3 モデルを表す三角形を描画するクラスを作成した.

```
rx = 0.866025*1.2
ry = 0.5*1.2
class tri:
    def __init__(self, x, y, r, color):
self.x = x
self.y = y
self.r = r
self.color = color

    def up(self, canvas):
x1 = self.x - rx * self.r
y1 = self.y + ry * self.r
x2 = x1 + self.r * 2
y2 = y1
x3 = self.x
y3 = self.y - self.r * 1.2
canvas.create_polygon(x1,y1,x2,y2,x3,y3, fill=self.color, outline='black')
```

```
    def down(self, canvas):
x1 = self.x - rx * self.r
y1 = self.y - ry * self.r
x2 = x1 + self.r * 2
y2 = y1
x3 = self.x
y3 = self.y + self.r * 1.2
canvas.create_polygon(x1,y1,x2,y2,x3,y3, fill=self.color, outline='black')
```

三角形を描画するオブジェクトはないが，多角形を描画する canvas.create_line() 関数があるため，これを使用する．canvas.create_line() 関数の引数は（1 つ目の頂点ｘ座標、1 つ目の頂点ｙ座標、2 つ目の頂点ｘ座標、2 つ目の頂点ｙ座標、....）というように書く．三角形を書くには 3 つの頂点が必要である [8]．

クラスには up と down 関数を作り，上向きの三角形と下向きの三角形を書くことができる．三角形の大きさは rx，ry で調整している．

```
def loop(argv, dev, init, fin):
    pos = np.zeros((int(n_atom), 3))
    draw_tilt_rect()
    for i in range(0,int(n_atom)):
m = lines[i+7].split('    ')
pos[i][0] = (float(m[0])-0.5)*x_lat+xp
pos[i][1] = (-float(m[1])+0.5)*y_lat+yp
pos[i][2] = float(m[2])
for j in range(-1,2):
    for k in range(-1,2):
if pos[i][2] >= init and pos[i][2] < init+dev:
    a = tri(pos[i][0]+x_lat*j, pos[i][1]+y_lat*k, rr,'black')
    a.down(canvas)
    draw_number(i, pos[i], argv,0,10)
```

```
elif pos[i][2] >= (fin-dev)/1.0 and pos[i][2] < (fin-0.01)/1.0:
    a = tri(pos[i][0]+x_lat*j, pos[i][1]+y_lat*k, rr,'white')
    a.up(canvas)
    draw_number(i, pos[i], argv,0,14)


root.after(10, loop(1,0.125, 0.0, 1.0))
root.mainloop()
```

POSCAR の原子数だけ for 文をまわす．原子数は n_atom に 40 が格納されているので 40 回ループする．for 文内では,「原子の座標をそれぞれ pos 配列に入れ，条件文にそって，三角形の向きと色を変えて表示する.」という操作を行なっている．

三角形の上下を描画すること，色を黒と白で区別化することで，別々の層を見分けれるようにしている．

別の層を描画するためには，loop 関数の変数を変更する必要がある．loop 関数の変数は下の表にまとめている．

layer	dev	init	fin
0-7 層	0.125	0.0	1.0
1-2 層	0.125	0.125	0.5-0.125
3-4 層	0.125	0.5-0.125	0.5+0.125
5-6 層	0.125	0.5+0.125	1.0-0.125

表 3.2: 各層ごとの変数.

粒界エネルギーの有限温度第一原理計算では，Einstein model のバネ定数決定に大量の第一原理計算が必要となる．ねじり角の違いにより，3x3 のモデルでは 40 原子であるが，5x5 では 104 原子，7x7 では 200 原子となる．100 原子程度のサイズの系の計算では，1 回あたり 10 分程度で済む．しかし，100 原子全てについて，4 種類のずれ，x,y,z の 3 方向，4 体積で計算するとなると，30 日程度必要になる [3]．そこで，原子への力のかかり方から同一の原子として扱えるものを等価なサイトと呼び，それらの計算結果も同じであるとすることによって計算時間を大幅に短縮する．図 3.9はねじり粒界の 0 層と 7 層の平面図であり，例えば 0 番と添えられたサイトは全て，原子の配置より等価なサイトであると確認できる．

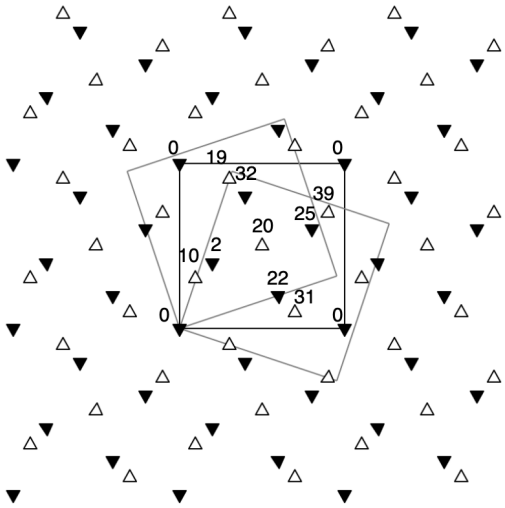

図 3.9: 0-7 層の平面図.

第4章　今後

今後の改善点として挙げられることは2つある．

1つ目は，プログラムを一つにまとめることである．現在，図3.1の黄色で囲われたプログラムの表示からアプリケーションの画面に戻ることができないため，SiteEnergy.pyとtwist07.pyをいちスクリーンとしてBoundaryViewer.py内での切り替えを行えるようにする必要がある．

2つ目は，アプリとしての機能を充実させることだ．例を挙げると，メニューバーや色の設定などのオプションをつけることである．POSCARを選択する画面を作り，表示する粒界の種類を増やしUI(User Interface)を快適にすることが挙げられる．テストをするという意味でも別のPOSCARを使用し，プログラムが正常に動作するか確認する必要がある．

第5章　まとめ

本研究では，Processing 言語で書かれた傾角粒界 (tilt) のエネルギー差のスペクトル表示とねじり粒界 (twist) のプログラムを Python でリライトし，独自のアプリケーション BoundaryViewer を開発した．Python で描画するために，開発には Python の標準ライブラリである Tkinter を用いている．

リライトする上で難点だったのは，Processing 言語には存在したモジュールが，Python にはなかったことであった．色の設定方法が RGB のカラーコードであることや図形の種類が少ないこと，これらを解決するためのプログラムを作成した．

自作したオブジェクトはリファクタリングでソースコードを整理することでただリライトするだけではなく，プログラムの中身をより理解することができた．

謝辞

本研究を行うにあたって，終始多大なる御指導，御鞭撻をいただいた西谷滋人教授に対し，深く御礼申し上げます．　また，本研究の進行に伴い，様々な助力，知識の供給を頂きました西谷研究室の同輩，並びに先輩方にご協力頂き，本研究を成就させることが出来ましたことを心から感謝の意を示します．　本当にありがとうございました．

参考文献

[1] 百合慶将, 「Al 粒界エネルギーへの添加元素の影響」, 関西学院大学卒業論文, (2022).

[2] 菊岡達也, 「結晶の圧縮・膨張のスペクトル表示」, 関西学院大学卒業論文, (2018).

[3] 天川純平,「Al ねじり粒界エネルギーの有限温度第一原理計算」, 関西学院大学卒業論文, (2021).

[4] Acro vision, 「Tkinterとは」, https://www.acrovision.jp/service/data/?p=616, (2023/1/25 accessed).

[5] めもぴー, 「Python Tkinter 色 (color) の設定」, https://memopy.hatenadiary.jp/entry/2017/06/11/092554, (2023/1/18 accessed).

[6] RGBとHSV･HSBの相互変換ツールと変換計算式, https://www.peko-step.com/tool/hsvrgb.html, (2023/1/18 accessed).

[7] 薫のHack,「Pythonで16進数を扱う」, https://kaworu.jpn.org/python/Pythonで16進数を扱う, (2023/1/18 accessed).

[8] とある科学の備忘録, 「canvas で図形を描画する」, https://shizenkarasuzon.hatenablog.com/entry/2018/12/31/080612, (2023/1/22 accessed).

[9] 複数の画面をボタン操作で切り替えて表示させる,
https://office54.net/python/tkinter/screen-change-tkraise, (2023/1/18 accessed).

[10] 浩羅, わかりやすいPythonの正規表現の例,
https://qiita.com/luohao0404/items/7135b2b96f9b0b196bf3, (2023/1/18 accessed).