さわって、うごかして理解する
フェーズフィールド法 ＜応用編＞

東京農工大学 山中晃徳

2022 年 11 月吉日

1　純物質の凝固現象のフェーズフィールドモデル

1.1　フェーズフィールドモデルの確認

固液界面の移動を追跡のために使用する非保存の秩序変数 ϕ の時間発展方程式は, Allen-Cahn 方程式より次式で与えられる.

$$\frac{\partial \phi}{\partial t} = M_\phi \left[\nabla(a^2 \nabla \phi) - \frac{\partial}{\partial x}\left(a \frac{\partial a}{\partial \theta} \frac{\partial \phi}{\partial y} \right) + \frac{\partial}{\partial y}\left(a \frac{\partial a}{\partial \theta} \frac{\partial \phi}{\partial x} \right) + 4W\phi(1-\phi)\left\{ \phi - 0.5 - \frac{15}{2W} \frac{L\,(T-T_m)}{T_m} \phi\,(1-\phi) + \chi \right\} \right] \tag{1}$$

ここで，M_ϕ はフェーズフィールドモビリティーである．L は潜熱を表し, T_m は融点を表す．また, χ は, 2 次アームと呼ばれるデンドライトに特有の組織形態を再現するために必要な化学的駆動力の微小な揺らぎである．勾配エネルギー係数 a は, 固液界面の法線方向に応じて, 次式により変化させる.

$$a(\theta) = \bar{a}\,[1 + \xi \cos\{k(\theta - \theta_0)\}] \tag{2}$$

ここで，$\bar{a}$ は勾配エネルギー係数，ξ は界面エネルギーの異方性の強さを表すパラメータ（異方性強度），k は界面エネルギーの異方性の対称性を表す係数，θ は固液界面の法線ベクトル（固相から液相に向く方向を正とする）と計算領域の水平方向との角度である．さらに, θ_0 は, デンドライトが優先的に成長する方向（角度）を決めるパラメータである.

凝固に伴う温度 T の変化を表す熱伝導方程式は, 次式で与えられる.

$$\frac{\partial T}{\partial t} = \kappa \nabla^2 T + 30\phi^2(1-\phi)^2 \frac{L}{c} \frac{\partial \phi}{\partial t} \tag{3}$$

ここで, c は比熱, κ は熱拡散係数であり, 次式のように熱伝導率 K と比熱 c の比として, $\kappa = K/c$ で与えられる.

純物質の凝固のフェーズフィールドモデルに含まれるパラメータとして, 勾配エネルギー係数に関する $\bar{a}$, エネルギー障壁 W およびフィールドモビリティー M_ϕ である．$\bar{a}$ と W は, 界面幅 δ と界面エネルギー γ と次式により関係付けられる.

$$\bar{a} = \sqrt{\frac{3\delta\gamma}{b}} \tag{4}$$

$$W = \frac{6\gamma b}{\delta} \tag{5}$$

ここで, b は, 界面幅を決めると求まる定数である. また, フィールドモビリティー M_ϕ は, 次式で与えられる.

$$M_\phi = \frac{bT_m\mu}{3\delta L} \tag{6}$$

ここで, μ は界面カイネティック係数と呼ばれる物性値である.

1.2 有限差分法による純物質凝固モデルの離散化

純物質の凝固のフェーズフィールドモデルにおける秩序変数 ϕ の時間発展方程式を有限差分法で離散化する. これは, 通常の Allen-Cahn 方程式の有限差分法を用いた離散化と同様である.

式 (1) で示した秩序変数 ϕ の時間発展方程式を再記する.

$$\begin{aligned}\frac{\partial\phi}{\partial t} = M_\phi \Bigg[& \nabla(a^2\nabla\phi) - \frac{\partial}{\partial x}\left(a\frac{\partial a}{\partial\theta}\frac{\partial\phi}{\partial y}\right) + \frac{\partial}{\partial y}\left(a\frac{\partial a}{\partial\theta}\frac{\partial\phi}{\partial x}\right) \\ & + 4W\phi(1-\phi)\left\{\phi - 0.5 - \frac{15}{2W}\frac{L\left(T-T_m\right)}{T_m}\phi\left(1-\phi\right) + \chi\right\}\Bigg]\end{aligned} \tag{7}$$

有限差分法で離散化するために, 上の式を次式のように記載する.

$$\begin{aligned}\frac{\partial\phi}{\partial t} = M_\phi \Bigg[& \nabla a^2\nabla\phi + a^2\nabla^2\phi + \frac{\partial A}{\partial x} + \frac{\partial B}{\partial y} \\ & + 4W\phi(1-\phi)\left\{\phi - 0.5 - \frac{15}{2W}\frac{L\left(T-T_m\right)}{T_m}\phi\left(1-\phi\right) + \chi\right\}\Bigg]\end{aligned} \tag{8}$$

ここで, $A = -a\frac{\partial a}{\partial\theta}\frac{\partial\phi}{\partial y}$ と $B = a\frac{\partial a}{\partial\theta}\frac{\partial\phi}{\partial x}$ とする.

次に, 式 (8) 左辺の時間微分を前進差分法で離散化すれば, 次式を得る.

$$\phi^{t+\Delta t}_{[i,j]} = \phi^t_{[i,j]} + M_\phi\left\{\nabla a^2\nabla\phi^t_{[i,j]} + a^2\nabla^2\phi^t_{[i,j]} + A' + B' + f\left(\phi^t_{[i,j]}\right)\right\}\Delta t \tag{9}$$

ここで, A と B は, それぞれ以下のように離散化する.

$$A = -a\frac{\partial a}{\partial\theta}\frac{\partial\phi}{\partial y} = -aa_0k\xi\sin\left(k\left(\theta^t_{[i,j]} - \theta_0\right)\right)\frac{\phi^t_{[i,j+1]} - \phi^t_{[i,j-1]}}{2\Delta y} \tag{10}$$

$$B = a\frac{\partial a}{\partial\theta}\frac{\partial\phi}{\partial x} = aa_0k\xi\sin\left(k\left(\theta^t_{[i,j]} - \theta_0\right)\right)\frac{\phi^t_{[i+1,j]} - \phi^t_{[i-1,j]}}{2\Delta x} \tag{11}$$

ここで, 固液界面の法線方向と計算領域の水平方向との角度 θ は, 次式のように計算できる.

$$\theta^t_{[i,j]} = \arctan\left(\frac{\frac{\partial\phi}{\partial y}}{\frac{\partial\phi}{\partial x}}\right) \tag{12}$$

式 (9) 右辺の駆動力項は, 次式のように離散化する.

$$f\left(\phi^t_{[i,j]}\right) = 4W\phi^t_{[i,j]}\left(1-\phi^t_{[i,j]}\right)\left\{\phi^t_{[i,j]} - 0.5 - \frac{15}{2W}\frac{L\left(T^t_{[i,j]} - T_m\right)}{T_m}\phi^t_{[i,j]}\left(1-\phi^t_{[i,j]}\right) + \chi\right\} \tag{13}$$

有限差分法による熱伝導方程式の離散化は, 拡散方程式の離散化とほとんど同じであり, 次式で与えられる.

$$\frac{T^{t+\Delta t}_{[i,j]} - T^{t}_{[i,j]}}{\Delta t} = \kappa \left(\frac{T^{t}_{[i+1,j]} - 2T^{t}_{[i,j]} + T^{t}_{[i-1,j]}}{\Delta x^2} + \frac{T^{t}_{[i,j+1]} - T^{t}_{[i,j]} + T^{t}_{[i,j-1]}}{\Delta y^2} \right) + 30 \left(\phi^{t}_{[i,j]} \right)^2 \left(1 - \phi^{t}_{[i,j]} \right)^2 \frac{L}{c} \frac{\phi^{t+\Delta t}_{[i,j]} - \phi^{t}_{[i,j]}}{\Delta t} \tag{14}$$

ここで，$T^{t+\Delta t}_{[i,j]}$ について整理し，$\Delta x = \Delta y$ を仮定すれば，次式が得られる.

$$\begin{aligned} T^{t+\Delta t}_{[i,j]} = T^{t}_{[i,j]} & \\ + \kappa \Delta t & \left(\frac{T^{t}_{[i+1,j]} + T^{t}_{[i-1,j]} + T^{t}_{[i,j+1]} + T^{t}_{[i,j-1]} - 4T^{t}_{[i,j]}}{\Delta x^2} \right) \\ + 30 & \left(\phi^{t}_{[i,j]} \right)^2 \left(1 - \phi^{t}_{[i,j]} \right)^2 \frac{L}{c} \left(\phi^{t+\Delta t}_{[i,j]} - \phi^{t}_{[i,j]} \right) \end{aligned} \tag{15}$$

次節では，式 (9) と式 (15) を用いて，純物資凝固モデルを数値計算するサンプルプログラムを説明する.

1.3 有限差分法による純物質凝固モデルの数値計算の Python プログラミング

純物質の凝固のフェーズフィールドモデルの数値計算の Python プログラミングは, 拡散方程式のものと同様であるため, 主なポイントのみ説明する.

■POINT 1　差分格子点の個数や差分格子点の間隔のほかに, 界面エネルギーの異方性強度や熱伝導率などの必要な定数を設定する． 異方性関数を使用する場合には, 2 行目の異方性強度 ξ は, $\xi \leq \frac{1}{k^2-1}$ を満たす値に設定する必要がある. 11 行目では, 式 (6) により, フェーズフィールドモビリティーを計算する.

Listing 1　計算条件や物性値の設定（抜粋）

```
1  gamma = 0.37 # 界面エネルギー[J/m2]
2  zeta = 0.03 # 異方性強度
3  aniso = 4.0 # 異方性モード数
4  angle0 = 0.0 # 優先成長方向 [radian]
5  T_melt = 1728.0 # 融点 [K]
6  K = 84.01 # 熱伝導率 [www/(mK)]
7  c = 5.42e+06 # 比熱 [J/K]
8  latent = 2.35e+09 # 潜熱[J/mol]
9  mu = 2.0 # 界面カイネティック係数[m/(Ks)]
10 kappa = K / c # 熱拡散係数
11 pmobi = b*T_melt*mu/(3.0*delta*latent) # モビリティー
12 T_0 = 1424.5 # 温度 [K]
```

■POINT 2　秩序変数 ϕ と温度 T の初期分布を設定する． 1 行目で, 固相の核の半径を設定する． 7-8 行目では, 計算領域の中心からの距離 r0 までの範囲に ϕ の初期分布を与える． また, 11 行目において, $T^{0}_{[i,j]} = T_0 + \phi\,(T_m - T_0)$ で初期温度を設定する.

Listing 2　ϕ と T の初期分布の設定

```
1  r0 = 3.*dx
```

```
2 for j in range(0,ny):
3     for i in range(0,nx):
4         phi[i,j] = 0.0
5         x = dx*(i-nx/2)
6         y = dy*(j-ny/2)
7         r = np.sqrt(x*x + y*y)
8         phi[i,j] = 0.5*(1.-np.tanh(np.sqrt(2.*www)/(2.*a0)*(r-r0)))
9         if phi[i,j] <= 1.0e-5:
10            phi[i,j] = 0.0
11        temp[i,j] = T_0 + phi[i,j] * (T_melt-T_0)
```

■POINT 3　全ての差分格子点上における, 秩序変数 ϕ と温度 T の空間 1 階微分や空間 2 階微分（ラプラシアン）を予め計算しておく関数 calcgrad を定義する．この関数では, 固液界面の法線方向を計算し, 勾配エネルギー係数やその微分も計算する．

1 行目では, 数値計算を高速化するために, numba ライブラリの JIT（Just In Time）コンパイラを用いる．9 ～ 16 行目では, ゼロノイマン境界条件を適用する．18 ～ 19 行目で, 秩序変数 ϕ の空間 1 階微分を計算する．20 ～ 21 行目では, 秩序変数 ϕ と温度 T の空間 2 階微分（ラプラシアン）を計算する．23 ～ 34 行目では, 秩序変数 ϕ の x 方向の 1 階微分の計算結果から固液界面の法線方向（計算領域に設定した x 軸からの回転角度 θ）を計算する．36 ～ 37 行目で, 勾配エネルギー係数 a とその θ による偏微分を計算する．最後に, 38 ～ 39 行目で, 式 (10) と式 (11) で与えた $-a\frac{\partial a}{\partial \theta}\frac{\partial \phi}{\partial y}$ と $a\frac{\partial a}{\partial \theta}\frac{\partial \phi}{\partial x}$ をそれぞれ計算する．

Listing 3　秩序変数と温度の空間微分および勾配エネルギー係数を計算する関数 calcgrad

```
1  @jit(nopython=True)
2  def calcgrad(phi,temp,zeta,a0,www,grad_phix,grad_phiy,lap_phi,lap_temp,ax,ay,a2):
3      for j in range(ny):
4          for i in range(nx):
5              ip = i + 1
6              im = i - 1
7              jp = j + 1
8              jm = j - 1
9              if ip > nx-1:
10                 ip = nx - 1
11             if im < 0:
12                 im = 0
13             if jp > ny-1:
14                 jp = ny - 1
15             if jm < 0:
16                 jm = 0
17
18             grad_phix[i,j] = (phi[ip,j]-phi[im,j])/(2.*dx)
19             grad_phiy[i,j] = (phi[i,jp]-phi[i,jm])/(2.*dy)
20             lap_phi[i,j] = (phi[ip,j]+phi[im,j]+phi[i,jp]+phi[i,jm]-4.*phi[i,j])/(dx*
                   dx)
21             lap_temp[i,j]= (temp[ip,j]+temp[im,j]+temp[i,jp]+temp[i,jm]-4.*temp[i,j])
                   /(dx*dx)
```

```
22
23              if grad_phix[i,j] == 0.:
24                  if grad_phiy[i,j] > 0.:
25                      angle = 0.5*pi
26                  else:
27                      angle = -0.5*pi
28              elif grad_phix[i,j] > 0.:
29                  if grad_phiy[i,j] > 0.:
30                      angle = np.arctan(grad_phiy[i,j]/grad_phix[i,j])
31                  else:
32                      angle = 2.0*pi + np.arctan(grad_phiy[i,j]/grad_phix[i,j])
33              else:
34                  angle = pi + np.arctan(grad_phiy[i,j]/grad_phix[i,j])
35
36              a = a0*(1. + zeta * np.cos(aniso*(angle-angle0)))
37              dadtheta = -a0*aniso*zeta*np.sin(aniso*(angle-angle0))
38              ay[i,j] = -a * dadtheta * grad_phiy[i,j]
39              ax[i,j] = a * dadtheta * grad_phix[i,j]
40              a2[i,j] = a * a
```

■POINT 4　式 (9) と式 (15) で示した秩序変数 ϕ と温度 T の時間発展方程式を計算する関数 timeevol を定義する.

1 行目で, 数値計算を高速化するために, numba ライブラリの JIT（Just In Time）コンパイラを用いる. 17 ～ 18 行目において, それぞれ $\frac{\partial}{\partial x}\left(-a\frac{\partial a}{\partial \theta}\frac{\partial \phi}{\partial y}\right)$ と $\frac{\partial}{\partial y}\left(a\frac{\partial a}{\partial \theta}\frac{\partial \phi}{\partial x}\right)$ を計算する. 20 ～ 21 行目では, $\frac{\partial a^2}{\partial x^2}$ と $\frac{\partial a^2}{\partial y^2}$ を計算する. 22 行目で, 凝固の駆動力を計算する. 23 行目では, $\frac{\partial a^2}{\partial x^2}\frac{\partial \phi}{\partial x}+\frac{\partial a^2}{\partial y^2}\frac{\partial \phi}{\partial y}$ を計算する. 25 ～ 27 行目において, 固液界面領域において, 化学的駆動力に加える微小な揺らぎを乱数を用いて計算する. 28 ～ 29 行目では, 式 (9) と式 (15) に基づき, 時刻 t での秩序変数 ϕ と温度 T から時刻 $t+\Delta t$ での値を計算する.

Listing 4　時間発展方程式を計算するサブルーチン

```
1  @jit(nopython=True)
2  def timeevol(phi,temp,zeta,a0,www,grad_phix,grad_phiy,lap_phi,lap_temp,ax,ay,a2):
3      for j in range(ny):
4          for i in range(nx):
5              ip = i + 1
6              im = i - 1
7              jp = j + 1
8              jm = j - 1
9              if ip > nx-1:
10                 ip = nx - 1
11             if im < 0:
12                 im = 0
13             if jp > ny-1:
14                 jp = ny -1
15             if jm < 0:
16                 jm = 0
```

```
17              dxdy = (ay[ip,j]-ay[im,j])/(2.*dx)
18              dydx = (ax[i,jp]-ax[i,jm])/(2.*dy)
19              grad_a2x = (a2[ip,j]-a2[im,j])/(2.*dx)
20              grad_a2y = (a2[i,jp]-a2[i,jm])/(2.*dy)
21              tet = phi[i,j]
22              drive = -latent * (temp[i,j]-T_melt) / T_melt
23              scal = grad_a2x*grad_phix[i,j]+grad_a2y*grad_phiy[i,j]
24
25              chi = 0.0
26              if tet > 0.0 and tet < 1.0:
27                  chi = np.random.uniform(-0.01,0.01)
28              phi[i,j] = phi[i,j] + (dxdy + dydx + a2[i,j]*lap_phi[i,j] + scal + 4.0*www
                    *tet*(1.0-tet)*(tet-0.5+15.0/(2.0*www)*drive*tet*(1.0-tet)+chi))*dt*
                    pmobi
29              temp[i,j] = temp[i,j] + kappa*lap_temp[i,j]*dt + 30.0*tet*tet*(1.0-tet)
                    *(1.0-tet)*(latent/c)*(phi[i,j]-tet)
```

■POINT 5　所定の時間ステップ数だけ時間発展方程式を数値計算し, 秩序変数 ϕ と温度 T の時間変化を計算するメインルーチンを記載する. ここでは, 3 行目と 4 行目で, それぞれ POINT 2 と POINT 3 で説明した関数を呼び出し, 秩序変数 ϕ と温度 T の時間変化を計算する. また, 所定の時間ステップ数おきに, 計算結果を matplotlib で出力する.

Listing 5　秩序変数 ϕ と温度 T の時間発展を計算するメインルーチン

```
1  start = time()
2  for nstep in range(stepmax+1):
3      calcgrad(phi,temp,zeta,a0,www,grad_phix,grad_phiy,lap_phi,lap_temp,ax,ay,a2)
4      timeevol(phi,temp,zeta,a0,www,grad_phix,grad_phiy,lap_phi,lap_temp,ax,ay,a2,phi_new,
           temp_new)
5      phi = phi_new
6      temp = temp_new
7
8      if nstep % 500 == 0:
9        print('step = ', nstep)
10       plt.figure(figsize=(12,6))
11       plt.rcParams["font.size"] = 15
12       plt.subplot(121)
13       plt.imshow(phi, cmap="bwr")
14       plt.title('Phase-field')
15       plt.colorbar()
16       plt.subplot(122)
17       plt.imshow(temp, cmap="bwr")
18       plt.title('Temperature [K]')
19       plt.colorbar()
20       plt.show()
21
```

```
end = time()
print("It takes", (end-start)*1000.0, "ms")
```

上記のサンプルプログラム Pure_Material_Solidification.ipynb を実行すると，計算領域に設置した, 円形状の固相（$\phi = 1$ の領域）の核が時間とともに拡大し, 界面エネルギーの異方性に起因して, 四角形に形状が変化する, 最終的には, 2 次アームも成長し, デンドライトが形成する.

2　多結晶粒成長のフェーズフィールドシミュレーション

マルチフェーズフィールドモデルの秩序変数の時間発展方程式は, 次式で与えられる.

$$\frac{\partial \phi_i}{\partial t} = -\frac{2}{n}\sum_{j=1}^{n} M_{ij}^{\phi}\left[\sum_{k=1}^{n}\left\{(W_{ik}-W_{jk})\phi_k + \frac{1}{2}(a_{ik}^2 - a_{jk}^2\nabla^2\phi_k\right\} + \frac{8}{\pi}\sqrt{\phi_i\phi_j}\Delta E_{ij}\right] \tag{16}$$

ここで, n は, ある空間座標に局所的に存在する粒の個数であり, 次式で求められる.

$$n = \sum_{\alpha=1}^{N} S_\alpha \tag{17}$$

ここで, N は系全体に含まれる結晶粒数である．また, S_α は, $\phi_\alpha > 0$ で 1，$\phi_\alpha = 0$ で 0 の値を取るステップ関数である．ΔE_{ij} は，粒 $i-j$ 間の界面移動の化学的駆動力である．

パラメータ M_{ij}^{ϕ}, W_{ij}, a_{ij} は，界面幅 δ および粒 $i-j$ 間の粒界エネルギー γ_{ij}，粒界モビリティ M_{ij} と次式により関係付けられる．

$$M_{ij}^{\phi} = \frac{\pi^2}{8\delta}M_{ij} \tag{18}$$

$$W_{ij} = \frac{4\gamma_{ij}}{\delta} \tag{19}$$

$$a_{ij} = \frac{2}{\pi}\sqrt{2\delta\gamma_{ij}} \tag{20}$$

ここで，$M_{ij} = M_{ji}$, $\gamma_{ij} = \gamma_{ji}$, $M_{ii} = 0$, $\gamma_{ii} = 0$ と考える．

2.1　有限差分法を用いた離散化

式 (16) の Allen-Cahn 方程式を有限差分法により離散化する． x 軸方向に N_x 個，y 軸方向に N_y 個の差分格子点を用い，時刻 t における格子点 $[l, m]$ ($l \in N_x$, $m \in N_y$) での秩序変数 ϕ_i を $\phi_{i[l,m]}^t$ と表記する．時間微分に 1 次精度前進差分，空間微分に 2 次精度中心差分を用いると，式 (16) の解くべき離散形として，

$$\phi_{i[l,m]}^{t+\Delta t} = \phi_{i[l,m]}^{t} - \frac{2}{n}\sum_{j=1}^{n} M_{ij}^{\phi}\left[\sum_{k=1}^{n}\left\{(W_{ik}-W_{jk})\phi_{k[l,m]}^{t} + \frac{1}{2}(a_{ik}^2 - a_{jk}^2)P_{i[l,m]}^{t}\right\} + \frac{8}{\pi}\sqrt{\phi_{i[l,m]}^{t}\phi_{j[l,m]}^{t}}\Delta E_{ij}\right] \tag{21}$$

$$P_{i[l,m]}^{t} = \nabla^2\phi_{i[l,m]}^{t} = \frac{\partial^2\phi_{i[l,m]}^{t}}{\partial x^2} + \frac{\partial^2\phi_{i[l,m]}^{t}}{\partial y^2} \simeq \frac{\phi_{i[l+1,m]}^{t} - 2\phi_{i[l,m]}^{t} + \phi_{i[l-1,m]}^{t}}{(\Delta x)^2} + \frac{\phi_{i[l+1,m]}^{t} - 2\phi_{i[l,m]}^{t} + \phi_{i[l-1,m]}^{t}}{(\Delta y)^2} \tag{22}$$

が得られる．ここで，$\Delta x = \Delta y$ の正方形構造格子を用いる場合，式 (22) は

$$P_{i[l,m]}^{t} = \nabla^2\phi_{i[l,m]}^{t} = \frac{\phi_{i[l+1,m]}^{t} + \phi_{i[l,m+1]}^{t} + \phi_{i[l-1,m]}^{t} + +\phi_{i[l,m-1]}^{t} - 4\phi_{i[l,m]}^{t}}{(\Delta x)^2} \tag{23}$$

とまとめることができる．離散式 (21) の Python プログラミング例を見ていこう．

2.2 Python プログラミング

多結晶粒成長に対する Allen-Cahn 方程式を数値的に解くサンプルプログラム multi-phase-field-model-for-grain-growth-2d.ipynb を説明する．このプログラムでは，7 個の秩序変数 ϕ_0,ϕ_1, ⋯, ϕ_6 を用いて，結晶粒 1 から結晶粒 6（秩序変数 ϕ_1からϕ_6）が母相（ϕ_0）を食って成長する多結晶粒成長を解析する．これまでの説明で理解できると考えるため, プログラム冒頭の「Python ライブラリのインポート」→「物性値や定数の設定」→「フェーズフィールドパラメータの計算」はここでも割愛し，それ以降のポイントを述べる．

■STEP 1　時間発展方程式 (21) の計算に用いる numby 配列 phi, phi_new, mf, nf を定義する．各配列の意味は，phi：時刻 t における各格子点での N 個の秩序変数値，mf：粒番号，nf：粒数，phi_new：時刻 $t+\Delta t$ における秩序変数値である．

Listing 6　配列の定義

```
phi = np.zeros((number_of_grain,nx,ny)) # phase-field variable at time t
phi_new = np.zeros((number_of_grain,nx,ny)) # phase-field variable at time t+dt
mf = np.zeros((15,nx,ny),dtype = int) # array for saving the grain IDs at the
    computational grid [i, j]
nf = np.zeros((nx,ny),dtype = int) # array for saving the number of grains at the
    computational grid [i, j]
```

■STEP 2　計算結果の可視化に必要な配列 gb, gi を定義する．gb は結晶粒界の可視化に用い，gi は各結晶粒を色分けして表示するのに用いる．これらの配列の具体的な使い方は STEP 7 において述べる．

Listing 7　配列 gb と gi の定義

```
gb = np.zeros((nx,ny)) # array for visualizing the grain boundaries
gi = np.zeros((nx,ny),dtype = int) # array for visualizing the grains by their IDs
```

■STEP 3　結晶粒 $i-j$ 間の粒界のフェーズフィールドパラメータ W_{ij}, a_{ij}, M_{ij}^{ϕ} および化学的駆動力 ΔE_{ij} を，配列 wij[i, j], aij[i, j], mij [i, j] および eij[i, j] にそれぞれ保存する．化学的駆動力 ΔE_{ij} は，母相（ϕ_0）とその他の結晶粒（ϕ_1 から ϕ_6）の間の粒界においてのみ非零とすることで，ϕ_1 から ϕ_6 が母相を浸食する条件としている．母相以外の結晶粒間の粒界では化学的駆動力が零のため，これらの粒界の移動は粒界エネルギーのみで駆動される．

Listing 8　パラメータおよび化学的駆動力の計算

```
for i in range(0,number_of_grain):
    for j in range(0,number_of_grain):
        wij[i,j] = www
        aij[i,j] = aaa
        mij[i,j] = pmobi
        eij[i,j] = 0.0
        if i == j:
            wij[i,j] = 0.0
            aij[i,j] = 0.0
```

```
10              mij[i,j] = 0.0
11          if i == 0 or j == 0:
12              eij[i,j] = eee
13          if i < j:
14              eij[i,j] = -eij[i,j]
```

■STEP 4　各格子点での結晶粒数 nf，結晶粒番号 mf を計算するためのサブルーチンを定義する．本プログラムでは, 境界条件は周期境界条件とした（8 行目から 15 行目）.

Listing 9　配列 mf と nf の計算

```
1  def update_nfmf(phi,mf,nf):
2      for m in range(ny):
3          for l in range(nx):
4              l_p = l + 1
5              l_m = l - 1
6              m_p = m + 1
7              m_m = m - 1
8              if l_p > nx-1:
9                  l_p = l_p - nx
10             if l_m < 0:
11                 l_m = l_m + nx
12             if m_p > ny-1:
13                 m_p = m_p - ny
14             if m_m < 0:
15                 m_m = m_m + ny
16             n = 0
17             for i in range(number_of_grain):
18                 if phi[i,l,m] > 0.0 or (phi[i,l,m] == 0.0 and phi[i,l_p,m] > 0.0 or
                       phi[i,l_m,m] > 0.0 or phi[i,l,m_p] > 0.0 or phi[i,l,m_m] > 0.0):
19                     n += 1
20                     mf[n-1,l,m] = i
21             nf[l,m] = n
```

■STEP 5　各格子点で時間発展方程式を計算し，$\phi^t_{i[l,m]}$ から $\phi^{t+\Delta t}_{i[l,m]}$ を求めるためのサブルーチンを定義する．計算の手順は,「境界条件の導入（8 行目から 15 行目）」→「n = nf[l,m] 個の秩序変数 $\phi^t_{i[l]}$ に対して時間発展方程式を解き，得られた $\phi^{t+\Delta t}_{i[l]}$ を p-new に保存（16 行目から 27 行目）」→「 $\phi^{t+\Delta t}_{i[l]}$ の値を，0 以上 1 以下かつ総和が 1 となるよう調整（29 行目から 35 行目）」→「配列 phi を phi_new で上書き（35 行目）」である.

Listing 10　時間発展方程式の計算

```
1  def update_phasefield(phi,phi_new,mf,nf,eij):
2      for m in range(ny):
3          for l in range(nx):
4              l_p = l + 1
5              l_m = l - 1
```

```
            m_p = m + 1
            m_m = m - 1
            if l_p > nx-1:
                l_p = l_p - nx
            if l_m < 0:
                l_m = l_m + nx
            if m_p > ny-1:
                m_p = m_p - ny
            if m_m < 0:
                m_m = m_m + ny
            for n1 in range(nf[l,m]):
                i = mf[n1,l,m]
                dpi = 0.0
                for n2 in range(nf[l,m]):
                    j = mf[n2,l,m]
                    ppp = 0.0
                    for n3 in range(nf[l,m]):
                        k = mf[n3,l,m]
                        ppp += (wij[i,k] - wij[j,k])*phi[k,l,m] + 0.5*(aij[i,k]**2 -
                            aij[j,k]**2)*(phi[k,l_p,m] + phi[k,l_m,m] + phi[k,l,m_p] +
                             phi[k,l,m_m] - 4.0*phi[k,l,m])/dx/dx
                        phii_phij = phi[i,l,m]*phi[j,l,m]
                        dpi = dpi - 2.0 * mij[i,j] / float(nf[l,m]) * (ppp - 8./pi*np.
                            sqrt(phii_phij)*eij[i,j])
                phi_new[i,l,m] = phi[i,l,m] + dpi *dt

    phi_new = np.where(phi_new <= 0.0,0.0,phi_new)
    phi_new = np.where(phi_new >= 1.0,1.0,phi_new)

    for m in range(ny):
        for l in range(nx):
            a = np.sum(phi_new[:,l,m])
            phi[:,l,m] = phi_new[:,l,m] / a
```

■STEP 6　母相の秩序変数 ϕ_0 および結晶粒 1 から結晶粒 6 の秩序変数 ϕ_1 から ϕ_6 の初期分布を作成する．まず，計算領域の全体を母相（$\phi_0 = 1$）としたのち，結晶粒 1 から結晶粒 6 の中心座標を乱数で決定し，それらの周囲の半径 $4\Delta x$ の領域を結晶粒内部（対応する秩序変数の値 $= 1$）とする．拡散界面領域では，秩序変数の平衡分布を用いて初期分布を設定する．

Listing 11　秩序変数の初期分布の設定

```
pphi = np.zeros((number_of_grain,nx,ny))
phi_new = np.zeros((number_of_grain,nx,ny))
mf = np.zeros((15,nx,ny),dtype = int)
nf = np.zeros((nx,ny),dtype = int)

```

```
phi[0,:,:] = 1.0
nf[:,:] = 1
r_nuclei = 3.*dx # radius of the initial grains

for i in range(1,number_of_grain):
    x_nuclei = int(rand()*nx)
    y_nuclei = int(rand()*ny)
    for m in range(ny):
        for l in range(nx):
            if l > nx-1:
                l = l - nx
            if l < 0:
                l = l + nx
            if m > ny-1:
                m = m - ny
            if m < 0:
                m = m + ny
            r = np.sqrt( (l *dx-x_nuclei*dx)**2 +(m*dy-y_nuclei*dy)**2 ) - r_nuclei
            tmp = np.sqrt(2.*www)/aaa*r
            phi_tmp = 0.5*(1.-np.sin(tmp))
            if tmp >= pi/2.:
                phi_tmp=0.
            if tmp <= -pi/2.:
                phi_tmp=1.
            if 0. < phi_tmp < 1.:
                nf_tmp = nf[l,m]+1
                nf[l,m] = nf_tmp
                mf[nf_tmp-1,l,m] = i
                phi[i,l,m] = phi_tmp
                phi[0,l,m] = phi[0,l,m]-phi[i,l,m]
                if phi[0,l,m] < 0.:
                    phi[0,l,m] = 0.
            if phi_tmp >= 1.:
                nf_tmp = 1
                nf[l,m] = nf_tmp
                mf[0,l,m] = i
                phi[i,l,m] = phi_tmp
                phi[0,l,m] = 0.
```

■STEP 7　以上で作成した秩序変数の初期分布を基に，初期の多結晶構造における結晶粒の分布（’grain ID’），粒界の分布（’grain boundary’），局所的な粒数 n = nf[l,m] の分布（’number of grains’）を描画する．grain ID は，各格子点 [l,m] で最大の値を取る秩序変数の添字番号を代表値として配列 gi[l,m] に格納し（7 行目から 11 行目），それらを 2 次元描画することで，各結晶粒を色分けして表示するものである．grain boundary は，各格子点 [l,m] で秩序変数の自乗和 $\sum \phi_i^2$ を計算して配列 gb[l,m] に格納し（5 行目），2 次元描画することで粒界を可視化する．$\sum \phi_i^2$ は結晶粒内で 1，粒界の中心で 1/2，三重点の中心で 1/3 のように，

領域ごとに異なる値を取るため，粒界形態の可視化に都合のよい指標としてしばしば用いられる．

Listing 12　初期多結晶構造の可視化

```
for m in range(0,ny):
    for l in range(0,nx):
        a = np.sum(phi[:,l,m])
        phi[:,l,m] = phi[:,l,m] / a
        gb[l,m] = np.sum(phi[:,l,m]*phi[:,l,m])
        phi_max = 0.
        for n in range(nf[l,m]):
            i = mf[n,l,m]
            if phi[i,l,m] > phi_max:
                gi[l,m] = i
                phi_max = phi[i,l,m]

fig = plt.figure(figsize=(7,4))
fig.set_dpi(120)
plt.subplots_adjust(wspace=0.3)

plt.subplot(1,3, 1)
plt.imshow(gi, cmap='jet',
           vmin=0, vmax=number_of_grain-1)
plt.title('grain ID')
plt.colorbar(aspect=20, pad=0.1, orientation='horizontal')
plt.subplot(1,3, 2)
plt.imshow(gb, cmap="gray",
           vmin=0.25, vmax=1.)
plt.title('grain boundary')
plt.colorbar(aspect=20, pad=0.1, orientation='horizontal')
plt.subplot(1,3, 3)
plt.imshow(nf, cmap='bwr',vmin=1, vmax=4)
plt.title('number of grains')
plt.colorbar(aspect=20, pad=0.1, orientation='horizontal')
plt.show()
```

■STEP 8　時間発展方程式を計算する関数を呼び出す処理を nsteps 回繰り返し，秩序変数の時間発展を順次計算する．また，500 ステップの繰り返しごとに，matplotlib を用いて結晶粒の分布（grain ID），粒界の分布（grain boundary）および局所的な粒数 n の分布（number of grains）を描画する．

Listing 13　時間発展方程式を計算するメインルーチン

```
for nstep in range(1,nsteps+1):
    update_nfmf(phi,mf,nf)
    update_phasefield(phi,phi_new,mf,nf,eij)

    if nstep % 50 == 0:
        print('nstep = ', nstep)
```

```
        for m in range(0,ny):
            for l in range(0,nx):
                gb[l,m] = np.sum(phi[:,l,m]*phi[:,l,m])
                phi_max = 0.
                for n in range(nf[l,m]):
                    i = mf[n,l,m]
                    if phi[i,l,m] > phi_max:
                        gi[l,m] = i
                        phi_max = phi[i,l,m]

        fig = plt.figure(figsize=(7,4))
        fig.set_dpi(120)
        plt.subplots_adjust(wspace=0.3)
        plt.subplot(1,3, 1)
        plt.imshow(gi, cmap='jet', vmin=0, vmax=number_of_grain-1)
        plt.title('grain ID') (以下省略)
```

上記のサンプルプログラム multi-phase-field-model-for-grain-growth-2d.ipynb を実行すると, 結晶粒 1 から結晶粒 6 が化学的駆動力により母相 ($\phi_0 = 1$) を浸食して成長し，200 step までに母相が完全に消滅することがわかる．母相の消滅後，残った結晶粒間の化学的駆動力は零である（STEP 3 参照）ため，粒界エネルギーのみに依存した純粋な粒界曲率駆動の粒成長が生じる．この過程では，大きな結晶粒が小さな結晶粒を食って成長する．さらに長時間の計算を行えば，最終的に 1 つの結晶粒のみが生き残る単結晶化まで再現することが可能である．