入門講座

Javaによる実践的科学技術プログラミング（Ⅲ）
―マルテンサイト変態のPhase-fieldシミュレーション―

小　山　敏　幸*

3.1 は　じ　め　に

最終回である今回は，マルテンサイト変態の2次元Phase-fieldシミュレーションプログラムについて説明する．今回も特定の合金系は設定せず，典型的なマルテンサイト変態の計算を行う．このプログラムは，各種の形状記憶合金のマルテンサイト変態や，誘電体の構造相転移における結晶変態等に活用できる．前回同様，読者は是非とも，ご自身のパソコンで計算を試みられたい．そして，マルテンサイトのドメイン形成がリアルタイムにて進行していく様子を楽しんでいただきたい．以下，計算手法について説明し，その後，ソースコードとともにプログラムの解説を行う．

3.2 計　算　手　法

マルテンサイト変態前後の相の名を，それぞれα相およびβ相とし，それぞれの結晶構造を立方晶および正方晶とする．本計算では無拡散変態を扱うので濃度場は考慮しない．本計算で考慮する秩序変数は，組織内におけるβ相の存在確率（つまりβ相のphase field）である．Phase-field法[(1)-(4)]では，この独立な秩序変数を用いて全自由エネルギーを汎関数形式に定式化し，この全自由エネルギーが効率よく減少するように，その秩序変数の発展方程式（この場合はLandauタイプの非保存系における発展方程式）が定義される．この発展方程式を数値計算することによって，秩序変数の時間および空間発展（すなわち組織形成過程）が算出される．以下，計算に用いる全自由エネルギーの評価式について説明する．

3.2.1 全自由エネルギーの定式化

全自由エネルギーG_{sys}は，化学的自由エネルギーG_c，勾配エネルギーE_{surf}，および弾性歪エネルギーE_{str}の総和として，$G_{sys}=G_c+E_{surf}+E_{str}$にて与えられる．これら個々のエネルギーの計算式は，以下のようにまとめられる．

(1) 化学的自由エネルギー

組織内におけるβ相の存在確率を$s_i(\mathbf{r}, t)$，$(i=1, 2, 3)$とする．$\mathbf{r}$は組織内の位置ベクトルで，tは時間である．添え字のiは，β相の3つのバリアントに対応する（β相は正方晶であるので，単位胞のc軸方向に依存して3つのバリアントが存在する）．α相の存在確率は，$1-\sum_{i=1}^{3} s_i(\mathbf{r}, t)$である．この$s_i(\mathbf{r}, t)$が$\beta$相のphase fieldであり，変域は$0 \le s_i \le 1$である．ここでは，マルテンサイト変態を記述する化学的自由エネルギーを，ランダウ展開形式を用いて，

$$G_c(s_1, s_2, s_3)=\int_{\mathbf{r}} \Delta G_c \left\{ \frac{a}{2}(s_1^2+s_2^2+s_3^2)-\frac{b}{3}(s_1^3+s_2^3+s_3^3) + \frac{c}{4}(s_1^2+s_2^2+s_3^2)^2 \right\} d\mathbf{r} \qquad (1)$$

と表現する[(5)(6)]．ΔG_cはα相とβ相の自由エネルギー差で（ここでは$\Delta G_c>0$とする），$-\Delta G_c$がα相からβ相への変態の駆動力となる．式内のa, b, およびcは定数であるが，aを用いて，bとcは$b=3(a+4)$, $c=2(a+6)$と設定され，このように置くことによって，式（1）右辺の積分内は，$s_i=0, 1$において極小となり，$s_i=a/(2a+12)$において極大をとる．またこの極大におけるエネルギー障壁の大きさは，$\Delta G_c a^3(a+8)/[32(a+6)^3]$である．本講座では2次元計算を扱い，$s_3$は考慮しないので，式（1）より化学ポテンシャルは，

* (独)物質・材料研究機構新構造材料センター，組織・特性計算グループ　主幹研究員（〒305-0047 つくば市千現 1-2-1）
Practical Java Programming in Materials Science and Engineering（Ⅲ）―Phase-field Simulation of Martensitic Transformation―; Toshiyuki Koyama (National Institute for Materials Science, Structural Metals Center, Tsukuba)
Keywords: *Java application, Windows software, diffusion-less transformation, twin, Phase-field method, evolution equation*
2009年6月10日受理

$$\frac{\delta G_c}{\delta s_1}=\Delta G_c s_1\{a-bs_1+c(s_1^2+s_2^2)\}$$
$$\frac{\delta G_c}{\delta s_2}=\Delta G_c s_2\{a-bs_2+c(s_1^2+s_2^2)\} \quad (2)$$

と導かれる．またパラメータについては，本計算では $\Delta G_c=1000$[J/mol]および $a=10$ とおいた．

(2) 勾配エネルギー

次に勾配エネルギーについては，通常の Phase-field 法の取り扱いに従い，

$$E_{surf}=\frac{1}{2}\kappa_s\int_{\mathbf{r}}[(\nabla s_1)^2+(\nabla s_2)^2+(\nabla s_3)^2]d\mathbf{r} \quad (3)$$

を用いて計算する[1]．κ_s は勾配エネルギー係数で，本計算では定数：$\kappa_s=5.0\times10^{-15}$[J·m²/mol]とした．$E_{surf}$ の s_i による変分は，

$$\frac{\delta E_{surf}}{\delta s_1}=-\kappa_s\nabla^2 s_1,\quad \frac{\delta E_{surf}}{\delta s_2}=-\kappa_s\nabla^2 s_2 \quad (4)$$

にて与えられる[3]．

(3) 弾性歪エネルギー

マルテンサイト変態は固相変態であるので，組織形態の安定性に弾性歪エネルギーが非常に大きく影響する．議論を簡潔にするために等方弾性体を仮定する．2 次元計算では，バリアント p の変態歪 $\varepsilon_{ij}^{00}(p)$ は，$\varepsilon_{11}^{00}(1)=\varepsilon_{22}^{00}(2)=\eta_1$ および $\varepsilon_{22}^{00}(1)=\varepsilon_{11}^{00}(2)=\eta_2$ と定義され（他の $\varepsilon_{ij}^{00}(p)=0$），これより全体の変態歪[7][8]は，$s_i$ の関数として，

$$\varepsilon_{ij}^0(\mathbf{r})=\varepsilon_{ij}^{00}(1)s_1(\mathbf{r})+\varepsilon_{ij}^{00}(2)s_2(\mathbf{r}) \quad (5)$$

と表現される（$\varepsilon_{ij}^{00}(p)$ は位置の関数ではなく，$\varepsilon_{ij}^0(\mathbf{r})$ が位置の関数である点に注意）．また物体表面には外部応力 σ_{ij}^A のみが作用していると仮定する．一定外力 σ_{ij}^A の作用下における弾性歪エネルギーの基礎式[7][8]は，通常の弾性歪エネルギーに外力のポテンシャルエネルギーを加えて，

$$\begin{aligned}E_{str}&=\frac{1}{2}\int_{\mathbf{r}}C_{ijkl}\varepsilon_{ij}^{el}(\mathbf{r})\varepsilon_{kl}^{el}(\mathbf{r})d\mathbf{r}-\sigma_{ij}^A\int_{\mathbf{r}}\varepsilon_{ij}^c(\mathbf{r})d\mathbf{r}\\&=\frac{1}{2}\int_{\mathbf{r}}C_{ijkl}\{\bar{\varepsilon}_{ij}^c+\delta\varepsilon_{ij}^c(\mathbf{r})-\varepsilon_{ij}^0(\mathbf{r})\}\{\bar{\varepsilon}_{kl}^c+\delta\varepsilon_{kl}^c(\mathbf{r})-\varepsilon_{kl}^0(\mathbf{r})\}d\mathbf{r}\\&\quad-\sigma_{ij}^A\bar{\varepsilon}_{ij}^c\end{aligned} \quad (6)$$

と表現できる（この式には外力に起因するポテンシャルエネルギーも含まれているので，正確には弾性歪エネルギーではなく，Gibbs エネルギーを記すべきであるが，化学的自由エネルギーとの混乱を避けるために，ここでは弾性歪エネルギーと記す）．C_{ijkl} は弾性定数，$\varepsilon_{ij}^{el}(\mathbf{r})$ は弾性歪で，$\varepsilon_{ij}^c(\mathbf{r})$ は拘束歪である．$\varepsilon_{ij}^c(\mathbf{r})$ を $\varepsilon_{ij}^c(\mathbf{r})=\bar{\varepsilon}_{ij}^c+\delta\varepsilon_{ij}^c(\mathbf{r})$ のように空間平均値 $\bar{\varepsilon}_{ij}^c$ とそこからの変動量 $\delta\varepsilon_{ij}^c(\mathbf{r})$ に分ける（$\bar{\varepsilon}_{ij}^c=\int_{\mathbf{r}}\varepsilon_{ij}^c(\mathbf{r})d\mathbf{r}$ および $\int_{\mathbf{r}}\delta\varepsilon_{ij}^c(\mathbf{r})d\mathbf{r}=0$ である）．物体表面の境界条件として，物体表面が力学的に拘束されていない自由表面を想定すると，力学的平衡状態では，$\bar{\varepsilon}_{ij}^c$ は条件式 $\partial E_{str}/\partial\bar{\varepsilon}_{ij}^c=0$ から決定され，式（6）より $\bar{\varepsilon}_{kl}^c=C_{ijkl}^{-1}\sigma_{ij}^A+\bar{\varepsilon}_{kl}^0$ が得られる[1]．C_{ijkl}^{-1} は弾性コンプライアンス（C_{ijkl} の逆行列）である．また変態歪の空間平均値 $\bar{\varepsilon}_{kl}^0$ は，$\bar{\varepsilon}_{kl}^0\equiv\int_{\mathbf{r}}\varepsilon_{ij}^0(\mathbf{r})d\mathbf{r}$ にて定義される．

さて $\delta E_{str}/\delta s_p$ を算出しよう．式（6）より，2 次元計算では，

$$\begin{aligned}\frac{\delta E_{str}}{\delta s_p}&=\frac{\delta E_{str}}{\delta\varepsilon_{ij}^0}\frac{\partial\varepsilon_{ij}^0}{\partial s_p}=-C_{ijkl}\{\bar{\varepsilon}_{kl}^c+\delta\varepsilon_{kl}^c(\mathbf{r})-\varepsilon_{kl}^c(\mathbf{r})\}\frac{\partial\varepsilon_{ij}^0}{\partial s_p}\\&=C_{ijkl}[\varepsilon_{kl}^0(\mathbf{r})-\bar{\varepsilon}_{kl}^0-\delta\varepsilon_{kl}^c(\mathbf{r})-\varepsilon_{kl}^A]\varepsilon_{ij}^{00}(p)\\&=[\varepsilon_{11}^0(\mathbf{r})-\bar{\varepsilon}_{11}^0-\delta\varepsilon_{11}^c(\mathbf{r})-\varepsilon_{11}^A]\{(\lambda+2\mu)\varepsilon_{11}^{00}(p)+\lambda\varepsilon_{22}^{00}(p)\}\\&\quad+[\varepsilon_{22}^0(\mathbf{r})-\bar{\varepsilon}_{22}^0-\delta\varepsilon_{22}^c(\mathbf{r})-\varepsilon_{22}^A]\{\lambda\varepsilon_{11}^{00}(p)+(\lambda+2\mu)\varepsilon_{22}^{00}(p)\}\end{aligned} \quad (7)$$

となる．ここで，$\partial\varepsilon_{ij}^0/\partial s_p=\varepsilon_{ij}^{00}(p)$，$\bar{\varepsilon}_{kl}^c=C_{ijkl}^{-1}\sigma_{ij}^A+\bar{\varepsilon}_{kl}^0$，および $C_{ijkl}=\lambda\delta_{ij}\delta_{kl}+\mu(\delta_{ik}\delta_{jl}+\delta_{il}\delta_{jk})$ を用いた[7][8]．δ_{ij} はクロネッカーデルタである．また $\varepsilon_{kl}^A\equiv C_{ijkl}^{-1}\sigma_{ij}^A$ と定義した．λ と μ はラーメの定数である．次に $\delta\varepsilon_{ij}^c(\mathbf{r})$ の計算について説明する．まず秩序変数 $s_p(\mathbf{r})$ のフーリエ変換を，

$\tilde{s}_p(\mathbf{k})=\frac{1}{V}\int_{\mathbf{r}}s_p(\mathbf{r})\exp(-i\mathbf{kr})d\mathbf{r}$ と定義すると（V は物体の体積），変態歪 $\varepsilon_{ij}^0(\mathbf{r})$ のフーリエ変換[9]は，

$$\tilde{\varepsilon}_{ij}^0(\mathbf{k})=\varepsilon_{ij}^{00}(1)\tilde{s}_1(\mathbf{k})+\varepsilon_{ij}^{00}(2)\tilde{s}_2(\mathbf{k}) \quad (8)$$

にて与えられる．$\mathbf{k}$ はフーリエ空間における波数ベクトルである．$\delta\varepsilon_{kl}^c(\mathbf{r})$ は力学的平衡方程式（力のつりあい方程式）に基づき，

$$\begin{aligned}\delta\tilde{\varepsilon}_{ij}^c(\mathbf{k})=\left\{\delta_{ik}n_jn_l-\frac{n_in_jn_kn_l}{2(1-\nu)}\right\}\Big(&\frac{2\nu}{1-2\nu}\delta_{kl}\delta_{mn}+\delta_{km}\delta_{ln}\\&+\delta_{kn}\delta_{lm}\Big)\tilde{\varepsilon}_{mn}^0(\mathbf{k})\end{aligned} \quad (9)$$

にて計算される[7][8]．ν はポアソン比である（$\nu=\lambda/\{2(\lambda+\mu)\}$）．2 次元計算（平面歪問題）に対して，具体的に $\delta\tilde{\varepsilon}_{11}^c(\mathbf{k})$ と $\delta\tilde{\varepsilon}_{22}^c(\mathbf{k})$ を書き下すと，

$$\begin{aligned}\delta\tilde{\varepsilon}_{11}^c(\mathbf{k})&=\frac{1}{1-2\nu}\left[2(1-\nu)-n_1^2-\frac{\nu}{1-\nu}n_2^2\right]n_1^2\tilde{\varepsilon}_{11}^0(\mathbf{k})\\&\quad+\frac{1}{1-2\nu}\left[2\nu-\frac{\nu}{1-\nu}n_1^2-n_2^2\right]n_1^2\tilde{\varepsilon}_{22}^0(\mathbf{k})\\\delta\tilde{\varepsilon}_{22}^c(\mathbf{k})&=\frac{1}{1-2\nu}\left[2\nu-n_1^2-\frac{\nu}{1-\nu}n_2^2\right]n_2^2\tilde{\varepsilon}_{11}^0(\mathbf{k})\\&\quad+\frac{1}{1-2\nu}\left[2(1-\nu)-\frac{\nu}{1-\nu}n_1^2-n_2^2\right]n_2^2\tilde{\varepsilon}_{22}^0(\mathbf{k})\end{aligned} \quad (10)$$

が得られる．ここで $\mathbf{n}\equiv\mathbf{k}/|\mathbf{k}|$ である．$\delta\varepsilon_{ij}^c(\mathbf{r})$ は，$\delta\tilde{\varepsilon}_{ij}^c(\mathbf{k})$ の逆フーリエ変換を数値計算することによって得られる．弾性歪エネルギーの計算に用いたパラメータ値は，$\mu=1.235\times10^{11}$[Pa]，$\lambda=1.500\times10^{11}$[Pa]，$\eta_1=0.08$，$\eta_2=-0.04$，$\sigma_{11}^A=0$，$-1$[GPa]，$a_0=0.3563$[nm]である．$a_0$ は格子定数で，モル体積の計算に使用している．例えば結晶構造が fcc である場合を考えると，単位胞に原子は 4 個含まれているので，モル体積は $N_Aa_0^3/4$ にて計算される（N_A はアボガドロ数）．

3.2.2 発展方程式

発展方程式は，非保存場の発展方程式となるので，

$$\frac{\partial s_1}{\partial t}=-M_s\frac{\delta G_{sys}}{\delta s_1},\quad \frac{\partial s_2}{\partial t}=-M_s\frac{\delta G_{sys}}{\delta s_2} \quad (11)$$

を使用する．M_s はマルテンサイト変態の緩和係数である．

これより，無次元化した方程式は，

$$\frac{\partial s_1}{\partial(tM_sRT)}=-\frac{\delta(G_{\mathrm{sys}}/\mathrm{RT})}{\delta s_1},\quad \frac{\partial s_2}{\partial(tM_sRT)}=-\frac{\delta(G_{\mathrm{sys}}/\mathrm{RT})}{\delta s_2} \tag{12}$$

となる(M_s の次元が[J·s]$^{-1}$である点に注意)．エネルギーは RT にて，時間は M_sRT を用いて無次元化している．

以下，プログラムを示す．内容については，ソースコード内のコメントを参照していただきたい．

3.3 プログラムの説明

↓

```
//***[プログラム(MT2D_001.java)] ****************************
//***[インポート文] ********************
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class MT2D_001 extends Frame{

//***[グローバル変数] ********************************
    static int ND=128;                      //組織1辺の分割数
    static int IG=7;                        //2^IG=ND
    static int nd=ND;                       //計算領域の1辺の分割数
    static int ndm=ND-1;                    //計算領域の1辺の分割数-1
    static int nd2=ND/2;                    //計算領域の1辺の分割数/2
    static int ig=IG;                       //2^ig=ND
    static int width;                       //Window 全体の幅
    static int height;                      //Window 全体の高さ
    static int xwidth;                      //描画領域の幅
    static int yheight;                     //描画領域の高さ
    static int insetx;                      //Window の枠の幅(左右および下)
    static int insety;                      //Window の枠の幅(上)
    static double PI=3.141592;              //π
    static double RR=8.3145;                //ガス定数
    static double[][]s1h=new double[ND][ND]; //組織内の秩序変数データ s1 配列
    static double[][]s2h=new double[ND][ND]; //組織内の秩序変数データ s2 配列
    static Graphics g;                  //自由エネルギー曲線画面のグラフィックスオブジェクト
    static double time1;                    //計算時間(カウント数)
    static double temp;                     //温度[K]

//高速フーリエ変換のプログラムについては，文献(9)をご参照下さい．
    static double qs;                   //フーリエ変換(qs：-1)と逆フーリエ変換(qs：1)の区別
    static double[][]xr=new double[ND][ND]; //フーリエ変換の実数パートに使用する配列
    static double[][]xi=new double[ND][ND]; //フーリエ変換の虚数パートに使用する配列
    static double[]xrf=new double[ND];      //フーリエ変換の実数パートに使用する配列
    static double[]xif=new double[ND];      //フーリエ変換の虚数パートに使用する配列
    static double[]s=new double[ND];        //sin のテーブル
    static double[]c=new double[ND];        //cos のテーブル
    static int[]ik=new int[ND];             //ビット反転操作の配列

//***[コンストラクタ]**************************
    public MT2D_001(){
      xwidth=400; yheight=400;              //描画画面の横と縦の長さ(ピクセル単位)
      insetx=4; insety=30;                  //描画画面のふちの長さ
      width=xwidth+insetx*2;                //描画 Window 全体の横の長さ
      height=yheight+insetx+insety;         //描画 Window 全体の縦の長さ
      setSize(width, height);               //描画 Window のセット
      setBackground(Color.white);           //描画 Window の描画部の色を白に設定
      setVisible(true);                     //描画 Window を見えるようにする
      addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent e){System.exit(0);}
                                        //Window を閉じる時の操作(Window の右上×の設定)
      });
    }

//***[メインプログラム] ****************************************
    public static void main(String[]args)throws Exception{//例外処理は行わない

      MT2D_001 prog=new MT2D_001();         //MT2D_001のインスタンス prog を生成

      double[][]ec11=new double[ND][ND];    //拘束歪変動量
      double[][]ec22=new double[ND][ND];    //拘束歪変動量
      double[][]ep11h0=new double[ND][ND];  //変態歪
      double[][]ep22h0=new double[ND][ND];  //変態歪
      double[][]ep11qrh0=new double[ND][ND];//拘束歪変動量のフーリエ変換(実数部)
      double[][]ep11qih0=new double[ND][ND];//拘束歪変動量のフーリエ変換(虚数部)
      double[][]ep22qrh0=new double[ND][ND];//拘束歪変動量のフーリエ変換(実数部)
      double[][]ep22qih0=new double[ND][ND];//拘束歪変動量のフーリエ変換(虚数部)
      double[][]eta_s1=new double[4][4];    //バリアント 1 の変態歪
      double[][]eta_s2=new double[4][4];    //バリアント 2 の変態歪
      double[][]s1k_su=new double[ND][ND];  //勾配ポテンシャル
      double[][]s2k_su=new double[ND][ND];  //勾配ポテンシャル

      double s1, s2;                        //マルテンサイトの phase field
      double s1k_chem, s1k_str;             //化学ポテンシャル，弾性ポテンシャル
      double s2k_chem, s2k_str;             //化学ポテンシャル，弾性ポテンシャル
      double c11, c12, c44, lam0, mu0, nu0; //弾性定数
      double el_fac;                        //弾性定数規格化変数
      double ep11T, ep22T;                  //個々の歪成分の和
      double ep11_0, ep22_0;                //組織内の変態歪の平均値
      double ep11_a, ep22_a, ep12_a, ep21_a; //外力に起因する歪
      double sig11_a, sig22_a;              //外力
      double Z11ep, Z12ep, Z21ep, Z22ep;    //フーリエ変換時に使用する係数
      double sum11, sum22;
      double s1ddtt, s2ddtt;                //phase field の時間変化量
      double delt;                          //時間きざみ

      int i, j, k, l, ii=0, jj=0;           //整数
      int p, q, m, n;                       //整数
      int ip, im, jp, jm;                   //整数
      double a1, temp;                      //計算領域，温度
      double time1max;                      //最大時間(計算を止める際に使用)
      double b1, vm0, atom_n;               //規格化長さ，モル体積，単位胞内の原子数
      double smob;                          //マルテンサイト変態ダイナミクスの緩和係数
      double nx, ny, nxx, nyy, a1nn;        //逆空間の基本ベクトル，その自乗，ノルム

      double AA0, AA1, AA2, AA3;            //化学的駆動力定数
      double a1_c, b1_c, c1_c;              //格子定数
      double kappa_s1, kappa_s2;            //勾配エネルギー定数
      double ds_fac;                        //phase field の揺らぎの大きさ

//---- 各種パラメータ設定----------------------------------------------------------
      delt=0.1;                 //時間きざみ入力

      temp=500.0;               //温度[K]
      a1=250.0*1.0E-09;         //計算領域[m]
      b1=a1/nd;                 //差分ブロックの長さ[m]

      time1=-10.0;              //初期設定時間
      time1max=1.0+1.0e+07;     //計算時間の最大値

      smob=1.0;                 //マルテンサイト変態ダイナミクスの緩和係数
      ds_fac=0.01;              //phase field の揺らぎ係数

      AA0=1000.0;               //マルテンサイト変態の化学的駆動力[J/mol]
      AA0=AA0/RR/temp;          //無次元化
      AA1=10.0; AA2=3.0*AA1+12.0; AA3=2.0*AA1+12.0;  //化学的駆動力定数

      kappa_s1=5.0e-15;         //勾配エネルギー定数[Jm^2/mol]
      kappa_s1=kappa_s2=kappa_s1/RR/temp/b1/b1;  //無次元化

      a1_c=b1_c=c1_c=3.563e-10;                  //格子定数[nm]
      atom_n=4.0; vm0=6.02E23*a1_c*b1_c*c1_c/atom_n;
                                          //モル体積の計算(fcc を仮定)[m^3/mol]

//---- s1 場の変態歪の設定----------------------------------------------------------
      eta_s1[1][1]=0.08; eta_s1[2][2]=-0.04; eta_s1[3][3]=0.0;
      eta_s1[1][2]=eta_s1[2][1]=eta_s1[1][3]=eta_s1[3][1]
                  =eta_s1[2][3]=eta_s1[3][2]=0.0;

//---- s2 場の変態歪の設定----------------------------------------------------------
      eta_s2[1][1]=eta_s1[2][2]; eta_s2[2][2]=eta_s1[1][1]; eta_s2[3][3]=0.0;
      eta_s2[1][2]=eta_s2[2][1]=eta_s2[1][3]=eta_s2[3][1]
                  =eta_s2[2][3]=eta_s2[3][2]=0.0;

//---- 弾性定数(Ni の場合)----------------------------------------------------------
      el_fac=1.0E+11*vm0/RR/temp;
      c11=2.508*el_fac;
      c44=1.235*el_fac;
      c12=1.500*el_fac;
      //c12=c11-2.0*c44;
      lam0=c12; mu0=c44;        //ラーメの定数
      nu0=lam0/2.0/(lam0+mu0); //ポアソン比

//---- 外力の設定----------------------------------------------------------
      sig22_a=0.0;              //ここでは外力を 0 に設定
      ep11_a=-lam0/4.0/mu0/(lam0+mu0)*sig22_a;  //平面歪を想定
      ep22_a=(lam0+2.0*mu0)/4.0/mu0/(lam0+mu0)*sig22_a;
      ep12_a=ep21_a=0.0;

//*** phase field の初期場設定と，sin および cos テーブルの設定 **************
      prog.ini_field();         //phase field の初期場設定
      prog.table();     //フーリエ変換のための sin と cos のテーブルとビット変換配列の設定

//*** phase field の時間発展の計算 ***************************************
      while(time1<=time1max){

//---- phase field の表示----------------------------------------------------------
        if((((int)(time1)%50)==0)){prog.update_draw(g);}
                                  //カウント数が50の倍数おきに場を描画
        //if((((int)(time1)%100)==0)){prog.repaint();}

//---- phase field の保存----------------------------------------------------------
        //if((((int)(time1)%200)==0)){prog.datsave();}
                                  //カウント数が200の倍数おきに場を保存
        //if(time1==3000.0){prog.datsave();}
                                  //カウント数が3000の時に場を保存

//*** 勾配ポテンシャル ***************************
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){
            ip=i+1; im=i-1; jp=j+1; jm=j-1;
            if(i==ndm){ip=0;} if(i==0){im=ndm;}
            if(j==ndm){jp=0;} if(j==0){jm=ndm;}
            s1k_su[i][j]=-kappa_s1*(s1h[ip][j]+s1h[im][j]+s1h[i][jp]+s1h[i][jm]
                                   -4.0*s1h[i][j]);      //式(4)
            s2k_su[i][j]=-kappa_s2*(s2h[ip][j]+s2h[im][j]+s2h[i][jp]+s2h[i][jm]
                                   -4.0*s2h[i][j]);      //式(4)
          }
```

```
        }

//*** 変態歪場のフーリエ変換 epll  * * * * * * * * * * * * * * * * * * * * *
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){
            xr[i][j]=epllh0[i][j]=eta_s1[1][1]*s1h[i][j]+eta_s2[1][1]*s2h[i][j];
                                                                        //式(5)
            xi[i][j]=0.0;
          }
        }
        qs=-1.0; prog.rcfft();  //実空間からフーリエ空間への変換(qs<0)
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){
            epllqrh0[i][j]=xr[i][j]; epllqih0[i][j]=xi[i][j];
          }
        }
        epllqrh0[0][0]=epllqih0[0][0]=0.0;

//*** 変態歪場のフーリエ変換 ep22  * * * * * * * * * * * * * * * * * * * * *
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){
            xr[i][j]=ep22h0[i][j]=eta_s1[2][2]*s1h[i][j]+eta_s2[2][2]*s2h[i][j];
                                                                        //式(5)
            xi[i][j]=0.0;
          }
        }
        qs=-1.0; prog.rcfft();  //実空間からフーリエ空間への変換(qs<0)
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){
            ep22qrh0[i][j]=xr[i][j]; ep22qih0[i][j]=xi[i][j];
          }
        }
        ep22qrh0[0][0]=ep22qih0[0][0]=0.0;

//*** 変態歪場の平均値の算出 * * * * * * * * * * * * * * * * * * * * * * *
        sum11=sum22=0.0;
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){sum11+=epllh0[i][j]; sum22+=ep22h0[i][j];}
        }
        epll_0=sum11/nd/nd; ep22_0=sum22/nd/nd;

//*** 拘束歪変動量 ecll の計算  * * * * * * * * * * * * * * * * * * * * *
        for(i=0; i<=ndm; i++){
          if(i<=nd2-1){ii=i;} if(i>=nd2){ii=i-nd;}
          for(j=0; j<=ndm; j++){
            if(j<=nd2-1){jj=j;} if(j>=nd2){jj=j-nd;}
            alnn=Math.sqrt((double)ii*(double)ii+(double)jj*(double)jj);
            if(alnn==0.){alnn=1.;}
            nxx=(double)ii/alnn*(double)ii/alnn;
            nyy=(double)jj/alnn*(double)jj/alnn;
            z11ep=nxx*(2.0*(1.0-nu0)-nxx-nu0/(1.0-nu0)*nyy)/(1.0-2.0*nu0);
            z12ep=nxx*(2.0*nu0-nyy-nu0/(1.0-nu0)*nxx)/(1.0-2.0*nu0);
            xr[i][j]=z11ep*epllqrh0[i][j]+z12ep*ep22qrh0[i][j];  //式(10)
            xi[i][j]=z11ep*epllqih0[i][j]+z12ep*ep22qih0[i][j];  //式(10)
          }
        }
        qs=1.0; prog.rcfft();  //フーリエ空間から実空間への変換(qs>0)
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){ ecll[i][j]=xr[i][j];}
        }

//*** 拘束歪変動量 ec22 の計算  * * * * * * * * * * * * * * * * * * * * *
        for(i=0; i<=ndm; i++){
          if(i<=nd2-1){ii=i;} if(i>=nd2){ii=i-nd;}
          for(j=0; j<=ndm; j++){
            if(j<=nd2-1){jj=j;} if(j>=nd2){jj=j-nd;}
            alnn=Math.sqrt((double)ii*(double)ii+(double)jj*(double)jj);
            if(alnn==0.){alnn=1.;}
            nxx=(double)ii/alnn*(double)ii/alnn;
            nyy=(double)jj/alnn*(double)jj/alnn;
            z21ep=nyy*(2.0*nu0-nxx-nu0/(1.0-nu0)*nyy)/(1.0-2.0*nu0);
            z22ep=nyy*(2.0*(1.0-nu0)-nyy-nu0/(1.0-nu0)*nxx)/(1.0-2.0*nu0);
            xr[i][j]=z21ep*epllqrh0[i][j]+z22ep*ep22qrh0[i][j];  //式(10)
            xi[i][j]=z21ep*epllqih0[i][j]+z22ep*ep22qih0[i][j];  //式(10)
          }
        }
        qs=1.0; prog.rcfft();  //フーリエ空間から実空間への変換(qs>0)
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){ec22[i][j]=xr[i][j];}
        }

//*** ポテンシャルと発展方程式の計算  * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){
            s1=s1h[i][j];  s2=s2h[i][j];

//*** 化学ポテンシャルの計算 * * * * * * * * * * * * * * * * * * * * * * *
            s1k_chem=AA0*s1*(AA1-AA2*s1+AA3*(s1*s1+s2*s2));  //式(2)
            s2k_chem=AA0*s2*(AA1-AA2*s2+AA3*(s1*s1+s2*s2));  //式(2)

//*** 弾性ポテンシャルの計算 * * * * * * * * * * * * * * * * * * * * * * *
            epllT=epllh0[i][j]-epll_0-ecll[i][j]-epll_a;
            ep22T=ep22h0[i][j]-ep22_0-ec22[i][j]-ep22_a;

            s1k_str=epllT*((lam0+2.0*mu0)*eta_s1[1][1]+lam0*eta_s1[2][2])
              +ep22T*((lam0+2.0*mu0)*eta_s1[2][2]+lam0*eta_s1[1][1]);      //式(7)
            s2k_str=epllT*((lam0+2.0*mu0)*eta_s2[1][1]+lam0*eta_s2[2][2])
              +ep22T*((lam0+2.0*mu0)*eta_s2[2][2]+lam0*eta_s2[1][1]);      //式(7)

//*** phase field の時間発展の計算  * * * * * * * * * * * * * * * * * * * *
            s1ddtt=-smob*(s1k_chem+s1k_su[i][j]+s1k_str);      //式(12)
            s2ddtt=-smob*(s2k_chem+s2k_su[i][j]+s2k_str);      //式(12)
            s1h[i][j]=s1h[i][j]+(s1ddtt+ds_fac*(2.0*Math.random()-1.0))*delt;
            s2h[i][j]=s2h[i][j]+(s2ddtt+ds_fac*(2.0*Math.random()-1.0))*delt;

//---- s の変域(0<=s<=1)の補正-----------------------------------------------------------
            if(s1h[i][j]>=1.0){s1h[i][j]=1.0;} if(s1h[i][j]<=0.0){s1h[i][j]=0.0;}
            if(s2h[i][j]>=1.0){s2h[i][j]=1.0;} if(s2h[i][j]<=0.0){s2h[i][j]=0.0;}
          }
        }

//*** [時間増加] * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        timel=timel+1.0;
      }//while

//-------------------------------------------------------------------------------------
  System.out.printf
            (''¥n 終了しました．グラフの右上×をクリックして終了してください．¥n'');
}//main

// 以下はサブルーチンである．
//*** [phase field の初期設定] * * * * * * * * * * * * * * * * * * * * * *
    public void ini_field(){
      int i, j;
      double fac1;

      fac1=0.5;       //最大の初期揺らぎ
      for(i=0; i<=ndm; i++){
        for(j=0; j<=ndm; j++){
          //s1h[i][j]=fac1*Math.random(); s2h[i][j]=fac1*Math.random();
                                                                  //均一に核を置く場合
          if(Math.abs(j-nd2)<(nd/40)){s1h[i][j]=Math.random();
                                      s2h[i][j]=Math.random();} //中央に核を置く場合
        }
      }

    }

//*** [フーリエ変換のための sin と cos のテーブルと，ビット変換配列の設定] * * * * * * * *
    public void table(){
      int it, it1, it2, mc, mn;
      double q;

      q=2.0*PI/(double)nd;
      for(it=0; it<=nd2-1; it++){c[it]=Math.cos(q*(double)it);
        s[it]=Math.sin(q*(double)it);}

      ik[0]=0; mn=nd2; mc=1;
      for(it1=1; it1<=ig; it1++){
        for(it2=0; it2<=mc-1; it2++){ik[it2+mc]=ik[it2]+mn;}
        mn=mn/2; mc=2*mc;
      }
    }

//  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    public void update_draw(Graphics g){g=getGraphics(); paint(g);}

//*** [phase field の描画] * * * * * * * * * * * * * * * * * * * * * * * *
    public void paint(Graphics g){
      //g.clearRect(0, 0, width, height);  //Window をクリア

      int i, j, ii, jj;
      int icol, icol_r, icol_g, icol_b;
      double c_r, c_g, c_b;
      int ixmin=0, iymin=0, igx, igy, irad0;
      int ixmax=xwidth, iymax=yheight;
      double c, x, xmax, xmin, y, ymax, ymin, rad0;

      xmin=0.; xmax=1.;                                   //横軸の最小値，最大値
      ymin=0.; ymax=1.;                                   //縦軸の最小値，最大値
      rad0=1.0/(double)nd/2.0;                            //差分ブロックの長さの半分
      irad0=1+(int)(((double)ixmax-(double)ixmin)/(xmax-xmin)*rad0);
                                                          //rad0 のピクセル化

      System.out.printf(''%f ¥n'', timel);          //計算の繰返し回数を標準入出力に表示

      for(i=0; i<=nd; i++){
        for(j=0; j<=nd; j++){
          //phase field の位置座標(実際の値)
          x=1.0/(double)nd*(double)i+rad0;
          y=1.0/(double)nd*(double)j+rad0;
          //phase field の位置座標(スクリーン座標に変換)
          igx=(int)(((double)ixmax-(double)ixmin)*(x-xmin)/(xmax-xmin)
                +(double)ixmin);
          igy=(int)(((double)iymax-(double)iymin)*(y-ymin)/(ymax-ymin)
                +(double)iymin);

          //個々の差分ブロックの phase field 値
          ii=i; jj=j;
          if(i==nd){ii=0;} if(j==nd){jj=0;}               //周期的境界条件

          c_r= s1h[ii][jj];                               //s1 を赤
          c_g= s2h[ii][jj];                               //s2 を赤
          c_b=1.0-c_r-c_g;                                //変態前の相を青
          if(c_r>1.0){c_r=1.0;} if(c_r<0.0){c_r=0.0;}
          if(c_g>1.0){c_g=1.0;} if(c_g<0.0){c_g=0.0;}
          if(c_b>1.0){c_b=1.0;} if(c_b<0.0){c_b=0.0;}

          icol_r=(int)(255.*c_r); icol_g=(int)(255.*c_g); icol_b=(int)(255.*c_b);
                                                          //256階層に変換
          g.setColor(new Color(icol_r, icol_g, icol_b));  //差分ブロックの色を設定
          g.fillRect(insetx+igx-irad0,insety+igy-irad0, irad0*2, irad0*2);
                                                          //個々の差分ブロック描画
        }
      }
    }

//*** [1 次元高速フーリエ変換(FFT)]  * * * * * * * * * * * * * * * * * * *
    public void fft(){
      int ix, ka, kb, l2, lf, mf, n2, nf;
      double tj, tr;
```

```
        l2=1;
        for(lf=1; lf<=ig; lf++){
          n2=nd2/l2;
          for(mf=1; mf<=l2; mf++){
            for(nf=0; nf<=n2-1; nf++){
              ix=nf*l2; ka=nf+2*n2*(mf-1); kb=ka+n2;
              tr=xrf[ka]-xrf[kb]; tj=xif[ka]-xif[kb];
              xrf[ka]=xrf[ka]+xrf[kb]; xif[ka]=xif[ka]+xif[kb];
              xrf[kb]=tr*c[ix]+tj*qs*s[ix]; xif[kb]=tj*c[ix]-tr*qs*s[ix];
            }
          }
          l2=l2*2;
        }
      }

//***[2次元高速フーリエ変換(RC FFT)] ********************
      public void rcfft(){
        int i, ic, ir, j;

        for(ir=0; ir<=ndm; ir++){
          for(ic=0; ic<=ndm; ic++){xrf[ic]=xr[ir][ic]; xif[ic]=xi[ir][ic];}
          fft();
          for(ic=0; ic<=ndm; ic++){xr[ir][ic]=xrf[ik[ic]]; xi[ir][ic]=xif[ik[ic]];}
        }
        for(ic=0; ic<=ndm; ic++){
          for(ir=0; ir<=ndm; ir++){xrf[ir]=xr[ir][ic]; xif[ir]=xi[ir][ic];}
          fft();
          for(ir=0; ir<=ndm; ir++){xr[ir][ic]=xrf[ik[ir]]; xi[ir][ic]=xif[ik[ir]];}
        }
        if(qs>0.0){return;}
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){xr[i][j]=xr[i][j]/nd/nd; xi[i][j]=xi[i][j]/nd/nd;}
        }
      }

//***[データの保存] ****************************
      private void datsave() throws Exception{
        int i, j;
        //保存ファイル名を test.dat とする.
        PrintWriter outfile=new PrintWriter(                    //ファイルのオープン
          new BufferedWriter(new FileWriter("test.dat", true)));//追記

        outfile.println(time1);                                 //カウントの書き込み
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){
            outfile.println(s1h[i][j]);                         //phase field の書き込み
            outfile.println(s2h[i][j]);                         //phase field の書き込み
          }
        }
        outfile.close();                                        //ファイルのクローズ
      }

//***[データの読込み] ****************************
      private void datin() throws Exception{
        int i, j;
        String s_data;

        BufferedReader infile=new BufferedReader(new FileReader("ini000.dat"));
                                                                //ファイルのオープン

        s_data=infile.readLine();                               //文字列として読み込み
        time1=new Double(s_data).doubleValue();                 //文字を数値へ変換
        for(i=0; i<=ndm; i++){
          for(j=0; j<=ndm; j++){
            s_data=infile.readLine();                           //文字列として読み込み
            s1h[i][j]=new Double(s_data).doubleValue();         //文字を数値へ変換
            s_data=infile.readLine();                           //文字列として読み込み
            s2h[i][j]=new Double(s_data).doubleValue();         //文字を数値へ変換
          }
        }
        infile.close();                                         //ファイルのクローズ
      }

// ********************************************************
}//MT2D_001
//*** プログラム終了 ********************************
```

3.4 計 算 結 果

図1は上記のプログラムの実行結果で，500 K における等温マルテンサイト変態の2次元シミュレーションである(外力は0)．黒は立方晶，灰色と白が正方晶で，それぞれバリアントの異なるドメイン s_1 とドメイン s_2 を表している．正方晶のc軸方向は，灰色が横方向で，白が縦方向である．また正方晶比(c/a)は1よりも大きい(c軸方向に伸びた単位胞)．図1は(001)面で，横方向が[100]および縦が[010]方向である．t' は無次元化した時間である．相分解の初期状態(a)は立方晶であり，乱数を用いて図の中央に正方晶の核を置いている．初期において，約45°の方向に傾いた灰色と白の対のドメインが形成され(b)，時間の進行に伴いこの対が槍状に成長していく(c)，(d)．灰色と白の界面は双晶界面となっている．図1では右45°に傾いた灰色と白の対の分率が多く，左45°の対が徐々に消滅していく(e)，(f)．後期において右45°方向に傾いた双晶ラメラ組織となり，細い灰色ドメインが次第に消滅していくことがわかる(g)，(h)．

図2と**図3**は，上下方向に1 GPa の圧縮応力をかけた状

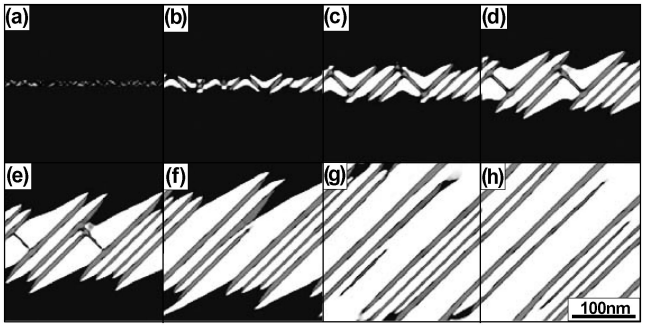

図1　c→t 構造相転移の2次元シミュレーション．
(a) $t'=0$, (b) $t'=20$, (c) $t'=40$, (d) $t'=60$, (e) $t'=80$, (f) $t'=100$, (g) $t'=140$, (h) $t'=200$.

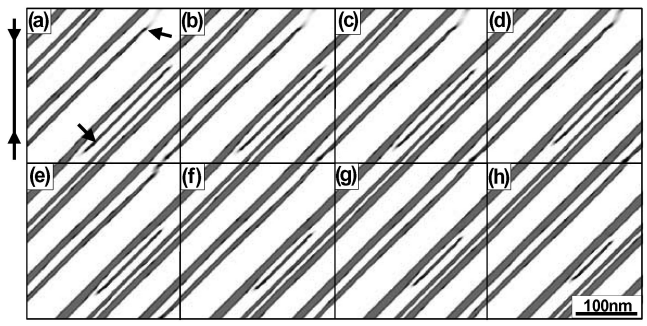

図2　外力下における組織変化(初期組織：図1(h)の場合)．
(a) $t'=0$, (b) $t'=10$, (c) $t'=20$, (d) $t'=30$, (e) $t'=40$, (f) $t'=50$, (g) $t'=60$, (h) $t'=70$.

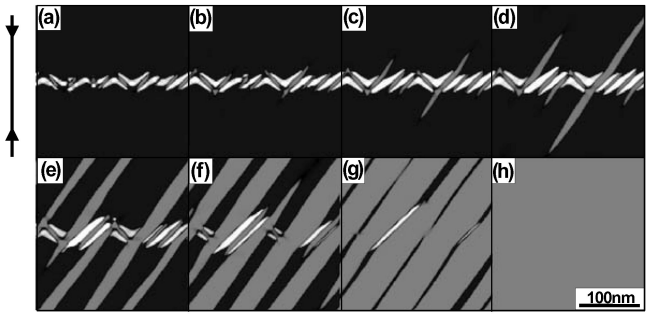

図3　外力下における組織変化(初期組織：図1(b)の場合)．
(a) $t'=0$, (b) $t'=10$, (c) $t'=20$, (d) $t'=30$, (e) $t'=40$, (f) $t'=50$, (g) $t'=60$, (h) $t'=70$.

態での組織変化で，それぞれ初期組織として図1(h)および図1(b)を用いた場合である．つまり図2はマルテンサト変態が完了した組織に応力を作用させた時の双晶ドメイン組織の変化過程で，図3はマルテンサイト変態の核形成初期の組織に外力を作用させた時のドメイン形成過程である．圧縮応力(1 GPa)を上下方向に作用させているので，c軸が横方向に伸びた単位胞を持つ灰色ドメインの方が白ドメインよりもエネルギー的に安定となる．また図2と図3の(a)−(h)の時間はそれぞれ等しい．図2では灰色ドメインが成長する部分((a)の上矢印)も観察されるが，反対に灰色ドメインから白ドメインに変化している部分((a)の下矢印)も存在する．しかし，組織全体の形態変化はほとんどなく，応力の影響は僅かである．一方，図3では，灰色ドメインの方が，白ドメインのよりも優先的に成長して，最終的に組織は灰色単一ドメインとなる．図2と図3の比較から，マルテンサイト変態初期の核形成段階に外部応力を作用させることが，単一ドメイン形成に効果的に働くことが理解できる[10]．

図2と図3を計算するプログラムは，3.3のプログラムを少し書き換えるだけで作成できる．また種々の解析作業を進めるためには，保存された計算データを読み出して組織を表示するプログラムや，個々の時間の秩序変数データを個別のファイルに取り出すプログラム(図2や図3の計算の入力データファイルを作成するときに必要)，組織計算結果の2次元数値データをjpg形式などのイメージファイルに変換して保存するプログラムなども必要である．これらのプログラムに関しては，紙面の関係で本講座では説明しないが，これら作業用プログラムも，本掲載プログラムのダウンロードページ（ http: / / www.nims.go.jp / mpsg / Phase − Field _ Modeling.htm)に公開している(なお，当該入門講座に関連して公開しているプログラム等について不具合・トラブルがあっても，著者および社団法人日本金属学会は責任を負いませんので，ご了承ください)．

3.5 おわりに

以上，今回はマルテンサイト変態のシミュレーションを例に，プログラムの説明を行った．本計算手法は，特にナノおよびメゾスケールにおけるマルテンサイト変態の微視的機構の本質解明に役立つと考えられる．また本計算から全歪(拘束歪)の平均値(局所的な拘束歪値の空間平均)も得られるので，たとえば周期的な外力に対し，全歪の平均値を計算すれば，形状記憶現象における応力−歪曲線を計算することができる．本計算手法は誘電体の構造相転移にも活用できるので，分極ドメイン組織形成や分極ヒステリシスにも応用できる[11]．したがって，各種センサー・アクチュエータ等の設計シミュレーションとしても，工学的に本計算手法は有用であろう．

また今回のプログラムには，高速フーリエ変換のサブルーチンが含まれている．したがって，このプログラムを改良することによって，パワースペクトルや自己相関を計算するプログラムを作成することもできる．また本稿の弾性場の計算はマイクロメカニクス[7][8]の数値計算になっているので，マイクロメカニクス分野の各種応用数値計算にも拡張することができる．本稿では，より基礎的な事項やさらに進んだ計算理論の詳細については紙面の関係上ふれなかったが，計算理論に関する基礎に関しては文献(1)−(3)等をご参照願いたい．また各種のデモプログラムやより進んだ計算理論の詳細については，著者のホームページ(http://www.nims.go.jp/mpsg/Phase−Field_Modeling.htm)にて公開しているので，興味のある方は参照していただきたい．

今回で本入門講座を終えるが，本稿を契機にパソコンで数値計算を行う学生さんが1人でも多く現れることを切に望む．

文　献

(1) 小山敏幸：ふぇらむ，**9**(2004), 240, 301, 376, 497, 905.
(2) 小山敏幸：まてりあ，**42**(2003), 397, 470.
(3) T. Koyama: Chapter 21 in *Springer Handbook of Materials Measurement Methods*, H. Czichos, T. Saito and L. Smith (Eds), Springer−Verlag, (2006), 1031−1054.
(4) T. Koyama: Science and Technology of Advanced Materials, **9**, 013006 (2008).
(5) Y. M. Jin, A. Artemev and A. G. Khachaturyan: Acta Mater., **49**(2001), 2309.
(6) T. Koyama and H. Onodera: Mater. Trans., **44**(2003), 2503.
(7) A. Khachaturyan: *Theory of Structural Transformations in Solids*., Wiley, New York, NY, (1983).
(8) T. Mura: *Micromechanics of Defects in Solids*, 2nd Rev. Ed., Kluwer Academic, (1991).
(9) 佐川雅彦，貴家仁志：高速フーリエ変換とその応用，昭晃堂，(1993).
(10) K. Tanaka, T. Ichitsubo and M. Koiwa: Mater. Sci. Eng., **A312**(2001), 118.
(11) T. Koyama and H. Onodera: Mater. Trans., **50**(2009), 970−976.

小山敏幸

★★★★★★★★★★★★★★★★★★★★★★

1988年3月　名古屋工業大学大学院　工学研究科　博士前期課程修了
1990年4月　名古屋工業大学大学院　工学研究科　博士後期課程単位取得後退学
1990年4月　名古屋工業大学　工学部　材料工学科　助手
2002年4月　㈱物質・材料研究機構　計算材料科学研究センター　主任研究員
2005年4月　㈱物質・材料研究機構　計算材料科学研究センター　主幹研究員
2006年4月　㈱物質・材料研究機構　計算科学センター　主幹研究員
2009年4月　㈱物質・材料研究機構　新構造材料センター　主幹研究員
現在に至る．
博士(工学)

専門分野：材料工学
主に材料組織形成の計算機シミュレーション研究に従事．

★★★★★★★★★★★★★★★★★★★★★★★★★★★★