入門講座

Javaによる実践的科学技術プログラミング(Ⅱ)
—拡散相分離のPhase-fieldシミュレーション—

小 山 敏 幸*

2.1 はじめに

今回は，先回のプログラムを修正して，拡散相分離のPhase-fieldシミュレーションプログラムを2種類作成する．1つは1次元濃度プロファイルの時間発展を計算するプログラムで，もう1つは2次元濃度場の時間発展プログラムである．いずれも合金系は相分離型の仮想的なA-B2元系を想定し，不規則相における2相分離の計算を行う．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．そして拡散現象が目の前にてリアルタイムで進行していく様子を楽しんでいただきたい．以下，まず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．

2.2 計算手法

2.2.1 A-B 2元系におけるα相の相分離の計算式

相分離する相をα相とする．本計算は，不規則A-B 2元系におけるα相の相分離が対象であるので，秩序変数は濃度場のみであり，したがって，本計算のPhase-field法[(1)-(4)]は，従来のCahn-Hilliardの非線形拡散方程式に基づく相分離の計算手法に等しい[(5)(6)]．ここでは独立な秩序変数としてB成分のモル分率を採用する．Phase-field法では，まず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最も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2元系の非線形拡散方程式)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[(1)]．以下まず，使用する全自由エネルギーの計算式について説明する．

(1) 全自由エネルギーの計算式

A-B 2元系のα相の原子1モル当たりの化学的自由エネルギーは，正則溶体近似に基づき，

$$G_c^{\alpha}(c,T)={}^{\circ}G_A^{\alpha}(T)(1-c)+{}^{\circ}G_B^{\alpha}(T)c+L_{A,B}^{\alpha}c(1-c)+RT\{c\ln c+(1-c)\ln(1-c)\} \quad (1)$$

にて与えられる[(7)]．cがB成分のモル分率である．組織形成過程の計算であるので，cは時間tおよび組織内の位置$\mathbf{r}=(x,y,z)$の関数として$c(\mathbf{r},t)$と表記するのが正しいが，ここでは式の煩雑さを避けるためにcと記す．${}^{\circ}G_x^{\alpha}(T)$は純成分X(=A, B)の自由エネルギーで，温度$T$の関数である．$R$はガス定数で，右辺最後の項が原子の配置のエントロピーに起因するエネルギー項である．右辺第3項が混合の過剰エンタルピー項で，係数の$L_{A,B}^{\alpha}$は相互作用パラメータと呼ばれる．$L_{A,B}^{\alpha}$は一般的には温度および組成の関数であるが，本計算では定数と仮定し$L_{A,B}^{\alpha}=25$[kJ/mol]とおいた．本計算は同一結晶構造内での等温時効における拡散相分解を対象とするので，自由エネルギーの基準の0として改めて，${}^{\circ}G_A^{\alpha}(T)(1-c)+{}^{\circ}G_B^{\alpha}(T)c$を採用しても計算結果は影響を受けない．(拡散相分離の計算では，拡散ポテンシャルの空間勾配が溶質移動の駆動力になる．拡散ポテンシャルを計算する際に自由エネルギーの組成に関する1次項は定数となり(温度一定)，さらに拡散ポテンシャルの空間勾配を計算する際に，この定数は消えてしまう)．したがって，ここでは，議論を簡単にするために，自由エネルギーの基準について，${}^{\circ}G_A^{\alpha}(T)={}^{\circ}G_B^{\alpha}(T)=0$と設定する．

次に界面エネルギーについて説明する．実際の界面エネルギーには，界面位置における平均場の化学的自由エネルギーも含まれるので，界面エネルギーと濃度勾配エネルギーは正確には異なるが，ここでは(界面エネルギー)≡(濃度勾配エネルギー)として話を進める．計算式は，

$$E_{\mathrm{surf}}=\frac{1}{2}\kappa_c(\nabla c_A)^2+\frac{1}{2}\kappa_c(\nabla c_B)^2$$

* (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 (Ⅱ) —Phase-field Simulation of Diffusional Phase Decomposition—;
Toshiyuki Koyama (National Institute for Materials Science, Structural Metals Center, Tsukuba)
Keywords: *Jave application, Windows software, phase decomposition, spinodal decomposition, Phase-field method, diffusion equation*
2009年5月11日受理

$$=\frac{1}{2}\kappa_c\{\nabla(1-c)\}^2+\frac{1}{2}\kappa_c(\nabla c)^2$$
$$=\kappa_c(\nabla c)^2 \qquad (2)$$

にて与えられる[1)6)]．スピノーダル分解の計算では濃度勾配エネルギー係数 κ_c は通常，方向に依存しない定数と仮定される．κ_c の値は，原子間相互作用パラメータ Ω，界面エネルギー密度 γ_s，界面幅 d_1，および相互作用距離[8] d_2 を用いて，近似的に見積もることが可能である[9]．本計算では，$\kappa_c=5.0\times10^{-15}$[J·m²/mol]と置いた．$\kappa_c$ の単位については，ここでは化学的自由エネルギーの単位[J/mol]との整合性から[J·m²/mol]を採用しているが，モル体積で割って[J/m]としている場合が多い点を記しておく．なお本講座はプログラミングに重点を置いているので，議論を簡単にするために弾性歪エネルギーは考慮しない．

以上から，全自由エネルギーは，化学的自由エネルギーと界面エネルギー(濃度勾配エネルギー)の和として，空間積分における汎関数形式にて，

$$G_{\rm sys}=\int_{\rm r}[L^{\alpha}_{A,B}c(1-c)+RT\{c\ln c+(1-c)\ln(1-c)\}+\kappa_{\rm c}(\nabla c)^2]d\mathbf{r} \qquad (3)$$

と表現される．$G_{\rm sys}$ の c による変分は，変分原理に基づき，

$$\frac{\delta G_{\rm sys}}{\delta c}=\frac{\partial G^{\alpha}_c}{\partial c}-2\kappa_{\rm c}\nabla^2 c \qquad (4)$$

にて与えられ[1)6)]，これは拡散ポテンシャルと呼ばれる．

(2) 発展方程式(非線形拡散方程式)の計算式

計算に用いる非線形拡散方程式[1]は，

$$\frac{\partial c}{\partial t}=\nabla\cdot\left(M_c(c,T)\nabla\frac{\delta G_{\rm sys}}{\delta c}\right) \qquad (5)$$

にて与えられる．$M_c(c, T)$は原子の拡散の易動度で，組成および温度の関数であるが[1]，合金組成 c_0 および温度 T の関数 $M_c(c_0, T)$ と近似する．式(4)を代入すると，

$$\frac{\partial c}{\partial t}=\nabla\cdot\left[M_c(c_0,T)\nabla\left\{\frac{\partial G^{\alpha}_c}{\partial c}-2\kappa_{\rm c}\nabla^2 c\right\}\right]$$
$$=\nabla\cdot\left[M_c(c_0,T)\left(\frac{\partial G^{\alpha}_c}{\partial c^2}\right)\nabla c\right]-2M_c(c_0,T)\kappa_{\rm c}\nabla^4 c \qquad (6)$$

を得る．ここで，$\tilde{D}\equiv M_c(c_0, T)(\partial^2 G^{\alpha}_c/\partial c^2)$および $\tilde{K}\equiv M_c(c_0, T)\kappa_c$ と置くと，Cahn–Hilliard の非線形拡散方程式

$$\frac{\partial c}{\partial t}=\nabla\cdot(\tilde{D}\nabla c)-2\tilde{K}\nabla^4 c \qquad (7)$$

に一致することがわかる．$\tilde{D}$ は相互拡散係数に他ならない．なお実際の相分離シミュレーションでは，式(4)の拡散ポテンシャル $\mu_{\rm sys}\equiv\delta G_{\rm sys}/\delta c$ を位置の関数 $\mu_{\rm sys}(\mathbf{r}, t)$として数値計算し，式(5)を

$$\frac{\partial c}{\partial t}=M_c\nabla^2\mu_{\rm sys} \qquad (8)$$

と変形して，差分法を用いて数値計算した方が効率的である．つまり組織内の任意の点における拡散ポテンシャル $\mu_{\rm sys}(\mathbf{r}, t)$を式(4)から求め，$\mu_{\rm sys}(\mathbf{r}, t)$について直接，差分計算を行う手法である．

最後に2次元計算を例に，拡散方程式の無次元化について説明する[9]．まずエネルギーは RT にて無次元化する．また2次元計算領域の1辺の長さを L，差分計算の分割数を N とすると，差分による空間分割セルの一辺の長さ b_1 は，$b_1=L/N$ である．この b_1 を用いて距離を無次元する(差分計算を容易にするため)．また時間は b_1^2/D にて無次元化される(D は拡散係数)．本計算では等温時効を想定しているので，易動度 $M_c(c_0, T)$は定数 M_c となる．また拡散係数と易動度の関係式(後述)より，M_c の次元は[m²/s]/[J/mol]である．以上より2次元(x, y)における非線形拡散方程式を，次元と合わせて表現すると，

$$\frac{\partial c}{\partial t}=M_c\left(\frac{\partial^2\mu_{\rm sys}}{\partial x^2}+\frac{\partial^2\mu_{\rm sys}}{\partial y^2}\right):\left[\frac{1}{s}\right]=\left[\frac{\rm m^2/s}{\rm J/mol}\right]\left[\frac{\rm J/mol}{\rm m^2}\right] \qquad (9)$$

となり，無次元化した方程式は，

$$\frac{\partial c}{\partial\left(\dfrac{t}{b_1^2/D}\right)}=\frac{M_cRT}{D}\left(\frac{\partial^2(\mu_{\rm sys}/RT)}{\partial(x/b_1)^2}+\frac{\partial^2(\mu_{\rm sys}/RT)}{\partial(y/b_1)^2}\right) \qquad (10)$$

となる．ここで濃度勾配エネルギー定数 κ_c は b_1^2RT にて無次元化される．

自己拡散係数 D^*_X(X=A, B)と $M_c(c, T)$には

$$M_c(c,T)=\left(\frac{D^*_B}{RT}(1-c)+\frac{D^*_A}{RT}c\right)c(1-c) \qquad (11)$$

の関係がある[1)6)]．D^*_X は固溶体中における X 原子の自己拡散係数であるので，正確には D^*_X は温度および濃度(媒質の固溶体の濃度)の関数であるが，ここでは最も単純に，$D^*_A=D^*_B=D$ と置き，かつ組成 c については合金組成 c_0 にて近似する．したがって，式(11)を変形することにより，式(10)右辺の係数部分が，$M_cRT/D=c_0(1-c_0)$と導かれる．実際に相分解シミュレーションを行う際に，拡散係数の値は，無次元化された計算時間 $t'\equiv t/(b_1^2/D)$を実時間 t に変換する時にのみ必要になる．無次元化された非線形拡散方程式は最終的に

$$\frac{\partial c}{\partial t'}=c_0(1-c_0)\left(\frac{\partial^2(\mu_{\rm sys}/RT)}{\partial(x/b_1)^2}+\frac{\partial^2(\mu_{\rm sys}/RT)}{\partial(y/b_1)^2}\right) \qquad (13)$$

であるので，無次元化された計算時間 t' にて議論する限りにおいては，拡散係数の値自体は特に必要ではない．なお本

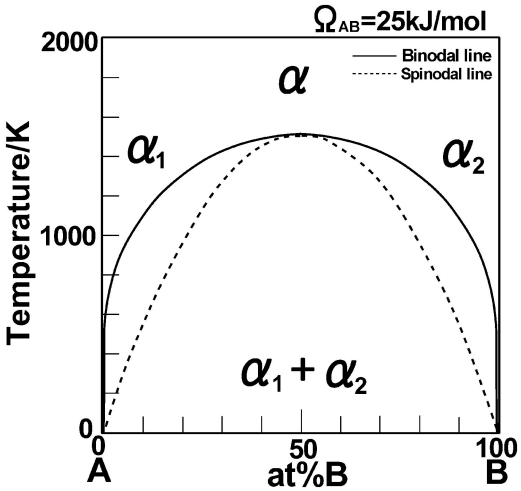

図1　計算に用いた A–B 2元系状態図．

計算では核形成–成長過程も計算するので，実際の計算では濃度揺らぎ項も乱数を用いて導入している．

(3) A–B 2 元系状態図

計算に用いた A–B 2 元系の状態図を**図 1** に示す．ここではα相のみを考慮している(液相は考慮していない)．実線がバイノーダル線，および点線がスピノーダル線である．相分離の計算は，高温の単相領域で溶体化した状態を初期状態とし，それを 2 相領域に持ちきたして等温時効した時の相分離過程を対象とする．時効温度は 1000 K とする．この温度における平衡組成とスピノーダル組成はそれぞれ，c_{Bi} = 0.07 および c_{sp} = 0.21，c_{Bi} = 0.93 および c_{sp} = 0.79 である．さて，プログラムの説明に入ろう．

2.3 プログラムの説明と計算例

基本的なプログラムの構造は，先回説明した F_curve に等しい．まず 1 次元濃度プロファイルの時間発展のプログラムを以下に示す．プログラム名は，PD1D_001.java である．プログラムの実行方法等は，先回の説明を参照していただきたい．さて先回同様，説明をプログラム内に記す．

↓

```
//***[プログラム(PD1D_001.java)] ************************
//***[インポート文] ********************
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class PD1D_001 extends Frame{

//***[グローバル変数] ********************************
    static int ND=512;                    //組織 1 辺の分割数
    static int nd=ND;                     //濃度の分割数
    static int ndm=ND-1;                  //濃度の分割数-1
    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[]ch=new double[ND];     //組織内の濃度データ配列
    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)
    static double c0;                     //合金組成(モル分率)

//***[コンストラクタ]*************************
    public PD1D_001(){
      xwidth=800; yheight=200;            //描画画面の横と縦の長さ(ピクセル単位)
      insetx=4; insety=30;                //描画画面の枠の長さ
      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

      PD1D_001 prog=new PD1D_001();       //PD1D_001のインスタンス prog を生成

      int i;                              //整数
      int ip, im;                         //整数(i+1, i-1)
      double delt;                        //時間刻み(無次元)
      double a1;                          //計算領域の 1 辺の長さ
      double timelmax;                    //計算時間(最大カウント数)
      double[]ck=new double[ND];          //拡散ポテンシャル
      double[]ch2=new double[ND];         //組織内の濃度データ予備配列
      double mu_chem, mu_surf;            //各ポテンシャル
      double c1, c2;                      //濃度(A : 1, B : 2)
      double L0;                          //原子間相互作用パラメータ
      double kappa_c;                     //濃度勾配エネルギー係数
      double Mc;                          //易動度
      double c_flu;                       //濃度場の揺らぎの大きさ
      double cddtt;                       //濃度の増分

      double b1;                          //差分ブロックサイズ
      double c2ip, c2im;                  //差分において c2 を中心に，その左右の濃度
      double sumc, dc;                    //濃度場の総和，平均組成からのずれ

//---- 各種パラメータ設定-----------------------------------------------------------
      temp=1000.0;                //[K]
      c0=0.4;                     //合金の平均組成(ここでは A-40 at%B 合金を設定している)
      delt=0.01;
      time1=0.0;                  //計算の繰返し回数
      timelmax=1.0e+08;           //繰返し回数の最大値

      a1=500.0;                   //2 次元計算領域の 1 辺の長さ(nm)
      a1=a1*1.0e-9;               //(m)に変換
      b1=a1/(double)nd;           //差分 1 ブロックのサイズ

      L0=2.5e+04;                 //原子間相互作用パラメータ(J/mol)
      L0=L0/RR/temp;              //無次元化

      kappa_c=5.0e-15;            //濃度勾配エネルギー係数，単位は[J m^2/mol]
      kappa_c=kappa_c/b1/b1/RR/temp;  //(b1^2*rr*temp)で無次元化
```

↗

```
      Mc=c0*(1.0-c0);             //拡散の易動度
      c_flu=0.1;                  //濃度ゆらぎ振幅の最大値

//---- 時間 0 における初期濃度プロファイル設定-------------------------------------
      prog.ini_comp_field();      //乱数によって生成する場合
      //prog.datin();             //ファイルから読み込む場合

//---- 濃度プロファイルの時間発展の計算-------------------------------------------
      while(time1<=timelmax){

//---- 濃度プロファイルの表示-----------------------------------------------------
        if((((int)(time1)%500)==0)){prog.repaint();}
                                  //カウント数が500の倍数おきに濃度プロファイル描画
//---- 濃度プロファイルの保存-----------------------------------------------------
        if((((int)(time1)%1000)==0)){ prog.datsave();}
                                  //カウント数が1000の倍数おきに濃度プロファイル保存
        //if(time1==3000.0){prog.datsave();}
                                  //カウント数が3000の時に濃度プロファイル保存

//---- 拡散ポテンシャルの計算-----------------------------------------------------
        for(i=0; i<=ndm; i++){
          ip=i+1; im=i-1;
          if(i==ndm){ip=0;} if(i==0){im=ndm;}          //周期的境界条件
          c2=ch[i]; c1=1.0-c2;                         //位置 i における c1 と c2
          c2ip=ch[ip]; c2im=ch[im];                    //差分においてc2の左右の濃度

          mu_chem=L0*(c1-c2)+Math.log(c2)-Math.log(c1); //化学ポテンシャル差
          mu_surf=-2.0*kappa_c*(c2ip+c2im-2.0*c2);      //濃度勾配のポテンシャル
          ck[i]=mu_chem+mu_surf;                        //拡散ポテンシャル
        }

//---- 濃度場の時間変化(非線形拡散方程式の差分解陽解法)---------------------------
        for(i=0; i<=ndm; i++){
          ip=i+1; im=i-1; if(i==ndm){ip=0;}if(i==0){im=ndm;}  //周期的境界条件
          cddtt=Mc*(ck[ip]+ck[im]-2.0*ck[i]);           //非線形拡散方程式
          //ch2[i]=ch[i]+cddtt*delt;                    //濃度場の時間発展
          ch2[i]=ch[i]+(cddtt+c_flu*(2.0*Math.random()-1.0))*delt;
                                                        //濃度場の時間発展(濃度揺らぎを考慮)
        }

//***[濃度場の収支の補正]*************************
//*** 数値計算であるので濃度場の収支の補正を行う(実際には毎ステップ行う必要はない) ***
        sumc=0.; for(i=0; i<=ndm; i++){sumc+=ch2[i];} //濃度プロファイルの積分
        dc=sumc/(double)nd-c0;                        //濃度プロファイルの変動量

        for(i=0; i<=ndm; i++){
          ch[i]=ch2[i]-dc;                            //濃度場の補正
          if(ch[i]>=1.){ch[i]=1.0-1.0e-6;}            //濃度が1を超えた場合の補正
          if(ch[i]<=0.){ch[i]=1.0e-6;}                //濃度が0を切った場合の補正
        }

//***[時間増加]*************************************

        time1=time1+1.0;                              //計算時間の増加
      }//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
//*** 初期濃度場設定 ***********************************
      public void ini_comp_field(){
        int i;
        double rnd0, fac1;

        fac1=0.01;                                    //初期濃度揺らぎの最大変化量を1%に設定
        for(i=0; i<=ndm; i++){
          rnd0=2.0*Math.random()-1.0; ch[i]=c0+rnd0*fac1;   //乱数にて初期濃度場を設定
        }
      }

//*** 濃度場描画 ***************************************
      public void paint(Graphics g){
        int i, ii, i1, ii1, i2, ii2;
        double x, xmax, xmin, dx, y, ymax, ymin, dy, d0;
        double c, cmax, cmin;
        double gx1, gy1, gx2, gy2;
        int ixmin, iymin, ixmax, iymax, igx1, igy1, igx2, igy2, id0;
        int idx, idy;
        double col;
```

↗

```
xmin＝0.; xmax＝1.; dx＝0.1;              //横軸の最小値，最大値，分割間隔(実際の値)
ymin＝0.; ymax＝1.; dy＝0.1;              //縦軸の最小値，最大値，分割間隔(実際の値)
cmin＝0.0; cmax＝1.0;                     //濃度の最小値，最大値(実際の値)
ixmin＝0; iymin＝0; ixmax＝xwidth;iymax＝yheight;
                                          //ピクセル単位の場合
idx＝(int)(0.1*ixmax);
idy＝(int)(0.1*iymax);

g.setColor(Color.white);                  //色を白に設定
g.fillRect(insetx, insety, xwidth, yheight);  //画面を上で指定した色で塗る
g.setColor(Color.black);                  //色を黒に設定
g.drawRect(insetx, insety, xwidth, yheight);  //グラフの外周を上で指定した色で描く

//グラフ内の縦横の線を等間隔に描く
for(i＝0; i＜＝ixmax; i＋＝idx){g.drawLine(insetx＋i, insety＋iymin,
                                          insetx＋i, insety＋iymax);}
for(i＝0; i＜＝iymax; i＋＝idy){g.drawLine(insetx＋ixmin, insety＋i,
                                          insetx＋ixmax, insety＋i);}
d0＝1.0/(double)nd/2.0;                   //差分ブロックの長さの半分
id0＝1＋(int)(((double)ixmax－(double)ixmin)/(xmax－xmin)*d0);
                                          //d0のピクセル化

System.out.printf(''%f ¥n'', time1);     //計算の繰返し回数を標準入出力に表示

g.setColor(Color.red);                    //色を赤に設定
for(i＝0; i＜＝ndm; i＋＋){
  //濃度プロファイルの値(実際の値)
  i1＝i; i2＝i＋1;
  gx1＝1./(double)nd*(double)i1＋d0; gx2＝1./(double)nd*(double)i2＋d0;
  ii1＝i1; ii2＝i2; if(i＝＝ndm){ii2＝0;}
  gy1＝(ch[ii1]－cmin)/(cmax－cmin); gy2＝(ch[ii2]－cmin)/(cmax－cmin);

  //濃度プロファイルの値をスクリーン座標に変換
  igx1＝(int)(((double)ixmax－(double)ixmin)*(gx1－xmin)/(xmax－xmin)
        ＋(double)ixmin);
  igy1＝(int)((double)iymin＋(double)iymax
        －(((double)iymax－(double)iymin)/(ymax－ymin)*(gy1－ymin)
        ＋(double)iymin));
  igx2＝(int)(((double)ixmax－(double)ixmin)*(gx2－xmin)/(xmax－xmin)
        ＋(double)ixmin);
  igy2＝(int)((double)iymin＋(double)iymax
        －(((double)iymax－(double)iymin)/(ymax－ymin)*(gy2－ymin)
        ＋(double)iymin));
    g.drawLine(insetx＋igx1, insety＋igy1, insetx＋igx2, insety＋igy2);
                                          //濃度プロファイルの描画
  }
 }

//***[データの保存]＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
 private void datsave()throws Exception{
  int i;

  //保存ファイル名を test.dat とする.
  PrintWriter outfile＝new PrintWriter(
   new BufferedWriter(new FileWriter
    (''test.dat'', true)));               //ファイルのオープン(追記)

  outfile.println(time1);                 //カウントの書き込み
  for(i＝0; i＜＝ndm; i＋＋){outfile.println(ch[i]);}
                                          //濃度場の書き込み
  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;
  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＋＋){
   s_data＝infile.readLine();             //文字列をして読み込み
   ch[i]＝new Double(s_data).doubleValue();  //文字を数値へ変換
  }
  infile.close();                         //ファイルのクローズ
 }

//  ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
}//PD1D_001
//*** プログラム終了 ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
```

本計算では，初期濃度プロファイルを乱数にて設定しているので，最後のサブルーチンは使用していない．**図2**はプログラム実行結果の1例で，A−40 at％B合金の1000 K等温時効における濃度プロファイルの時間発展である．初期に典型的なスピノーダル分解が生じ，時効の進行に伴い，濃度ピーク同士が互いに溶質を奪い合ってオストワルド成長していく様子が計算されている．

次に合金組成を変えて，同様に相分解させた時の濃度プロファイル(時間$t'＝3000$における結果)を**図3**に示す．状態図の中央組成付近では，典型的なスピノーダル分解組織となるが，合金組成が低下して，ほぼスピノーダル線上の$c_0＝0.2$では，濃度プロファイルの周期性が若干崩れてくる．合金組成がスピノーダル線とバイノーダル線の中央付近である$c_0＝0.15$では，まばらに濃度ピークが形成され，核形成−成長型相分離の特徴を示すようになる．最後に合金組成がバイノーダル線のすぐ内側の$c_0＝0.1$では，完全に核形成−成長型相分離の相分離となる．通常，スピノーダル分解と核形成−成長型相分解は，異なる相分離メカニズムとして説明される場合が多いが，原子の拡散の観点からは，基本的には1つの非線形拡散方程式から計算される単一の現象である[5]．ただし，図3の中央組成の濃度プロファイルと端の組成の濃度プロファイルの比較からわかるように，得られる組織形態は，スピノーダル線付近を境に大きく変化する．このことが，従来，スピノーダル分解と核形成−成長型相分解を分けて考えてきた所以であろう(もちろんスピノーダル点は，過飽和固溶体からの微小濃度揺らぎに対して，相分離に対する化学的駆動力が負から正に変わる点であり，相分離の本質的な機構は異なる)．

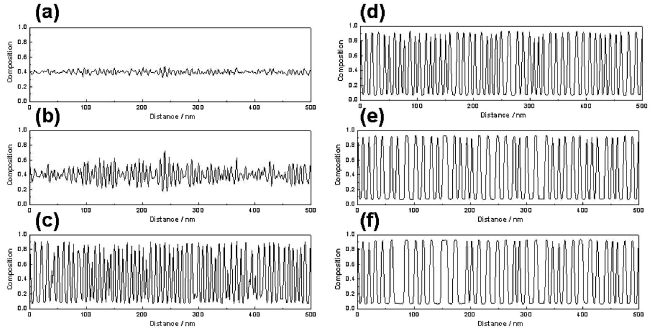

図2　A−40 at％B合金の1000 K等温時効における1次元濃度プロファイルの時間変化.
(a) $t'＝10$, (b) $t'＝20$, (c) $t'＝50$, (d) $t'＝200$, (e) $t'＝1000$, (f) $t'＝3000$.

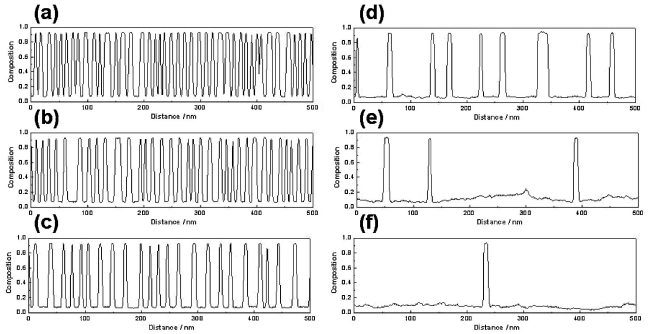

図3　A−B合金の1000 K等温時効($t'＝3000$)における1次元濃度プロファイルの合金組成による変化.
(a) A−50 at％B, (b) A−40 at％B, (c) A−30 at％B, (d) A−20 at％B, (e) A−15 at％B, (f) A−10 at％B.

さて次に，このプログラムを2次元に拡張してみよう．説明は以下のようになる（プログラム名はPD2D_001.java）．先の濃度プロファイルの計算プログラムと比較しながら見ていただきたい．

↓

```
//***［プログラム（PD2D_001.java）］ * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//***［インポート文］* * * * * * * * * * * * * * * * * * * * * * * * * * * *
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class PD2D_001 extends Frame{

//***［グローバル変数］ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    static int ND＝64;                          //組織1辺の分割数
    static int nd＝ND;                          //濃度の分割数
    static int ndm＝ND－1;                      //濃度の分割数－1
    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[][]ch＝new double[ND][ND];    //組織内の濃度データ配列
    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)
    static double c0;                           //合金組成(モル分率)

//***［コンストラクタ］* * * * * * * * * * * * * * * * * * * * * * * * *
    public PD2D_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;                    //描画画面の枠の長さ
      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)
      });
    }

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

      PD2D_001 prog＝new PD2D_001();        //PD2D_001のインスタンス prog を生成

      int i, j;                             //整数
      int ip, im, jp, jm;                   //整数(i＋1, i－1, j＋1, j－1)
      double delt;                          //時間刻み(無次元)
      double a1;                            //計算領域の1辺の長さ
      double timelmax;                      //計算時間(最大カウント数)
      double[][]ck＝new double[ND][ND];     //拡散ポテンシャル
      double[][]ch2＝new double[ND][ND];    //組織内の濃度データ予備配列
      double mu_chem, mu_surf;              //各ポテンシャル
      double c1, c2;                        //濃度(Fe：1, Cu：2)
      double L0;                            //原子間相互作用パラメータ
      double kappa_c;                       //濃度勾配エネルギー係数
      double Mc;                            //易動度関数とその微分
      double c_flu;                         //濃度場の揺らぎの大きさ
      double cddtt;                         //濃度の増分

      double b1;                            //差分ブロックサイズ
      double c2ip, c2im, c2jp, c2jm;//差分ブロックにおいて c2 を中心に，その上下左右の濃度
      double sumc, dc;                      //濃度場の総和，平均組成からのずれ

//----  各種パラメータ設定----------------------------------------------------------------
      temp＝1000.0;             //[K]
      c0＝0.4;                  //合金の平均組成(ここでは A－40 at%B 合金を設定している)
      delt＝0.04;               //時間きざみ
      time1＝0.0;               //計算の繰返し回数
      timelmax＝1.0e＋08;       //繰返し回数の最大値

      a1＝60.0;                 //2次元計算領域の1辺の長さ(nm)
      a1＝a1*1.0e－9;           //(m)に変換
      b1＝a1/(double)nd;        //差分1ブロックのサイズ

      L0＝2.5e＋04;             //原子間相互作用パラメータ(J/mol)
      L0＝L0/RR/temp;           //無次元化

      kappa_c＝5.0e－15;        //濃度勾配エネルギー係数，単位は[J m^2/mol]
      kappa_c＝kappa_c/b1/b1/RR/temp;
                                //濃度勾配エネルギー係数，(b1^2*rr*temp)で無次元化

      Mc＝c0*(1.0－c0);         //拡散の易動度
      c_flu＝0.1;

//----  時間 0 における初期濃度場設定----------------------------------------------------
      prog.ini_comp_field();
      //prog.datin();           //ファイルから読み込む場合

//----  濃度場の時間発展の計算------------------------------------------------------------
      while(time1＜＝timelmax){

//----  濃度場の表示----------------------------------------------------------------------
        //カウント数が200の倍数おきに濃度場描画
        if((((int)(time1)%200)＝＝0)){prog.update_draw(g);}
                                                  //描画時のチラツキを抑えるため
        //if((((int)(time1)%200)＝＝0)){prog.repaint();}
```

↗

```
//----  濃度場の保存----------------------------------------------------------------------
        if((((int)(time1)%500)＝＝0)){prog.datsave();}
                                                  //カウント数が500の倍数おきに濃度場保存
        //if(time1＝＝3000.0){prog.datsave();}    //カウント数が3000の時に濃度場保存

//----  拡散ポテンシャルの計算------------------------------------------------------------
        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;}    //周期的境界条件
            if(j＝＝ndm){jp＝0;} if(j＝＝0){jm＝ndm;}    //周期的境界条件

            c2＝ch[i][j];  c1＝1.0－c2;                 //位置(i, j)における c1 と c2
            c2ip＝ch[ip][j]; c2im＝ch[im][j];
            c2jp＝ch[i][jp]; c2jm＝ch[i][jm];       //差分において c2 の前後左右の濃度

            mu_chem＝L0*(c1－c2)＋Math.log(c2)－Math.log(c1);
                                                    //化学ポテンシャル差
            mu_surf＝－2.0*kappa_c*(c2ip＋c2im＋c2jp＋c2jm－4.0*c2);
                                                    //濃度勾配のポテンシャル
            ck[i][j]＝mu_chem＋mu_surf;             //拡散ポテンシャル
          }
        }

//----  濃度場の時間変化(非線形拡散方程式の差分解陽解法)----------------------------------
        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;}    //周期的境界条件
            if(j＝＝ndm){jp＝0;}  if(j＝＝0){jm＝ndm;}    //周期的境界条件
            cddtt＝Mc*(ck[ip][j]＋ck[im][j]＋ck[i][jp]＋ck[i][jm]－4.0* ck[i][j]);
                                                    //非線形拡散方程式
            //ch2[i][j]＝ch[i][j]＋cddtt*delt;      //濃度場の時間発展
            ch2[i][j]＝ch[i][j]＋(cddtt＋c_flu*(2.0*Math.random()－1.0)) *delt;
                                                    //濃度場の時間発展(濃度揺らぎ導入)
          }
        }

//***［濃度場の収支の補正］ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*** 数値計算であるので濃度場の収支の補正を行う(実際には毎ステップ行う必要はない).］ * *
        sumc＝0.;
        for(i＝0; i＜＝ndm; i＋＋){
          for(j＝0; j＜＝ndm; j＋＋){
            sumc＝sumc＋ch2[i][j];                  //濃度場の積分
          }
        }
        dc＝sumc/(double)nd/(double)nd－c0;         //濃度場の変動量

        for(i＝0; i＜＝ndm;i＋＋){
          for(j＝0; j＜＝ndm;j＋＋){
            ch[i][j]＝ch2[i][j]－dc;                //濃度場の補正
            if(ch[i][j]＞＝1.){ch[i][j]＝1.0－1.0e－6;}    //濃度が1を超えた場合の補正
            if(ch[i][j]＜＝0.){ch[i][j]＝1.0e－6;}         //濃度が0を切った場合の補正
          }
        }

//***[時間増加]  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        time1＝time1＋1.0;                          //計算時間の増加
      }//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．
//*** 初期濃度場設定 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    public void ini_comp_field(){
      int i, j;
      double rnd0, fac1;

      fac1＝0.01;                           //初期濃度揺らぎの最大変化量を1%に設定
      for(i＝0; i＜＝ndm;i＋＋){
        for(j＝0; j＜＝ndm;j＋＋){
          rnd0＝2.0*Math.random()－1.0;
          ch[i][j]＝c0＋rnd0*fac1;                  //乱数にて初期濃度場を設定
        }
      }

    }

//***  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//チラツキ防止のためdraw関数を別に定義
    public void update_draw(Graphics g){ g＝getGraphics(); paint(g);}

//*** 濃度場描画  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    public void paint(Graphics g){
      //g.clearRect(0, 0, width, height);         //Window をクリア

      int i, j, ii, jj;
      double c, x, xmax, xmin, y, ymax, ymin, rad0;
      int ixmin＝0, iymin＝0, igx, igy, irad0;
      int ixmax＝xwidth, iymax＝yheight;
      int ico1;

      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'', time1);          //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++){
            //濃度場の位置座標(実際の値)
            x=1.0/(double)nd*(double)i+rad0;
            y=1.0/(double)nd*(double)j+rad0;
            //濃度場の位置座標(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濃度値
            ii=i; jj=j;
            if(i==nd){ii=0;} if(j==nd){jj=0;}          //周期的境界条件
            icol=(int)(255.0*(1.0-ch[ii][jj]));        //色の諧調をグレースケールにする
            //icol=(int)(255.0*ch[ii][jj]);            //明暗を反転する場合
            if(icol>=255){icol=255;}if(icol<=0){icol=0;}
                                                       //明暗の範囲の補正
            g.setColor(new Color(icol, icol, icol));   //濃度を明暗で設定
            g.fillRect(insetx+igx-irad0, insety+igy-irad0, irad0*2, irad0*2);
                                                       //個々の差分ブロック描画
          }
        }
      }

//***[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(new BufferedWriter
            (new FileWriter(''test.dat'', true)));     //ファイルのオープン追記
        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(ch[i][j]);                 //濃度場の書き込み
          }
        }
        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();                      //文字列をして読み込み
        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();                  //文字列をして読み込み
            ch[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
      }

//***  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
}//PD2D_001
//***プログラム終了 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

このプログラムを実行すると，2次元濃度場の時間発展がリアルタイムにてパソコン画面で見ることが出来る．**図4**に計算結果の濃度場の時間発展を示す．図4(a)～(h)は，合金組成 $c_0=0.4$(A-40 at%B)の時効温度1000 Kにおけるスピノーダル分解の時間発展である．図中の相分解組織の明暗が局所的なB成分の組成に対応し，白～黒が0 at%B～100 at%Bを表している．また計算領域1辺は60 nmに設定してある．

初期状態は過飽和固溶体(最大で±1%程度の濃度揺らぎを乱数によって与えてある)であり，相分解の初期に均一な“まだら構造”が形成され[(a)-(d)]，その後，組織は孤立した微細な析出粒子が分散する組織となり[(e)-(g)]，時効の進行に伴いオストワルド成長によって析出相が粗大化していく[(g)-(h)]．合金組成や，時効温度を変えて種々の条件下での相分解過程を，パソコン上にてその場観察しながら計算を進めることができる．

さて以上は，相分離過程を計算するプログラムであるが，実際に各種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，濃度プロファイルや濃度場を表示するプログラム，個々の時間の濃度プロファイル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のような図を書く時に必要)，また濃度場の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，これら作業用のプログラムも，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)に公開するので，活用していただけると嬉しい(ただし公開しているプログラム等に関して不具合・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)．

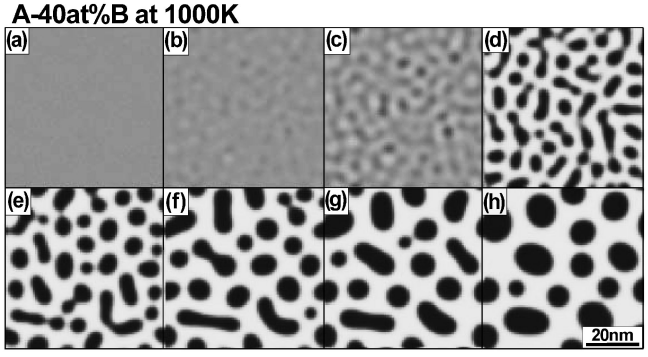

図4　A-40 at%B合金の1000 K等温時効における2次元相分離シミュレーション．
(a) $t'=0$, (b) $t'=10$, (c) $t'=20$, (d) $t'=50$,
(e) $t'=100$, (f) $t'=200$, (g) $t'=300$, (h) $t'=500$.

2.4 おわりに

以上，A-B2元系の拡散相分離シミュレーションを例にとり，パソコン上で簡潔に相分解シミュレーションを行うプログラムについて説明した．本プログラムは，学部や大学院における材料組織学の授業においても効果的に活用できると思う．最近では，多くの授業でPowerPointによる説明が行われるので，その場で，濃度プロファイルや濃度場の時間発展をデモンストレーションすることは教育効果としても有効であろう．もちろん，本プログラムを少し改良するだけで，現実の2成分系相分離のプログラムを作成することができる．例えば，Fe-CrやFe-Cuのフェライト相のスピノーダル分解プログラム[9]は比較的容易に作成可能である(ただしGibbsエネルギー内の磁気項の取り扱いは若干複雑である)．また本計算手法は連続体モデルに基づく解析であるので，合金に限らず，セラミクスやポリマーアロイの相分離の計算にも適用できる利点がある．特に近年，種々の分野で活発に研究が進められているナノ・メゾスケールにおける材料設計では，デバイスのスケール自体が微視的な材料組織のスケールに重なってくるので，組織形態の解析が，これまでに

も増して重要となりつつある．この時，本稿のようなプログラミングのスキルを身につけておくことは，次世代の新材料・デバイス設計において，有力な武器になるのではないだろうか．

なお本稿では，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．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．次回は，マルテンサイト変態の2次元シミュレーションプログラムについて解説する予定である．

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**(2008), 013006.
（5）T. Miyazaki, A. Takeuchi, T. Koyama and T. Kozakai: Trans. JIM, **32**(1991), 915–920.
（6）J. E. Hilliard: Phase Transformation, ed. by H. I. Aaronson, ASM, Metals Park, Ohio, (1970), 497.
（7）N. Saunders and A. P. Miodownik: CALPHAD, Pergamon, (1998).
（8）J. W. Cahn: The Selected Works of J. W. Cahn, ed. by W. C. Carter and W. C. Johnaon, TMS, (1998), 29.
（9）小山敏幸：ふぇらむ，**11**(2006), 647.

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．

★★★★★★★★★★★★★★★★★★★★★★★★★★★★