入門講座

Javaによる実践的科学技術プログラミング（Ⅰ）
―数値計算，グラフの描画，およびデータの入出力―

小　山　敏　幸*

1.1 は　じ　め　に

本連載講座は通常のJavaの解説ではない点を，あらかじめお断りしておく．従来，Java関連の書籍では，文字列を“Hello World”と表示するところから始まり，クラスの概念，インスタンスの生成，継承，･･･等，オブジェクト指向のプログラミングの考え方が展開される場合が多い．Javaを用いてソフトウェア(ごく簡単なものも含めて)を作成することを目的とするならば，このような教科書は有用である．しかし，技術者・研究者で，ソフトウェア作成ではなく，単純に数値計算がしたいだけであるならば，上記のテキストの大部分は必要のない知識である．なぜならば，通常，技術者・研究者が計算において必要とする操作の大部分は，以下の3種類のみであるからである．

⑴　手続き的な数値の計算
⑵　結果の可視化(グラフや画像)
⑶　数値データの保存と読み出し

きちんとしたソフトウェア作成自体を目的としなければ，プログラムにおいて，ボタン等のコンポーネントの作成や過度の例外処理などは必要ない．単にソースプログラムが書けて，それをコンパイルし，上記3つの操作に対して，プログラムを実行し，結果さえ得られれば良い(計算結果の数値データさえ保存できれば，それを綺麗にグラフ化・可視化するソフトは巷にあふれている)．プログラムに入力値が必要な場合には，極論すれば，ソースコードに毎回直接書き込み，その都度コンパイルして実行しても，ほとんどの場合，日常必要とする数値計算に支障は起きない．さらにプログラムも，実行するたびにその都度強制終了するものであってもかまわないのである．

Javaに限った話ではないが，基本的に通常のWindowsソフトウェアを作成することを目的としてしまうと，イベントドリブン型の部分，すなわち，ボタンやツールーバー，テキストボックス，クリック，ドラッグ，･･･など，本来の数値計算部分とは全く無関係な部分を作りこまなくてはならなくなるために，Windowsプログラミングは非常に複雑になる(コンポーネントウェアの概念にてプログラミングの手間はかなり削減されてはいるが，根本的に数値計算以外の部分が多く煩雑である点にかわりはない)．数値計算をしたいだけであるならば，ボタンやツールーバー，クリック，ドラッグなどのイベントドリブン型の部分の知識は全く必要ないので，この部分の学習は後回しでかまわない(後々，きちんとしたWindowsソフトウェアを作成したくなった時にあらためて勉強すれば良い)．

もう1つ最近のパソコンの高性能化をあらためて考えてみていただきたい．現在のパソコンは，10年前の通常の大型計算機並みの能力がある．10年前，数千万単位であったマシンが，現在は数万円で個人的で購入できるのである(実験機器でたとえるならば，10年前の最新鋭の電子顕微鏡が，現在，個人で買えるようになったようなものである)．これを研究・教育に使わないのは，明らかにもったいない．もちろん，既存のソフトウェアは豊富であり，文書作成や研究発表などでパソコンは大いに活用されている．しかし，みずからソースコードを書いて数値計算する用途には，あまり使用されていないのではないだろうか．本当は誰でも簡単にプログラミングして数値計算できるのである．

そこで，本解説では，上記の3つの操作に限定して，非常に簡単にJavaアプリケーションを作成する手法を解説する．なおJavaアプレットについては言及しない．Javaアプレットには，インターネット上で稼動するソフトウェアを簡

* ㈱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 (I) —Numerical Calculation, Graph Drawing and Data Input/Output—;
Toshiyuki Koyama (National Institute for Materials Science, Structural Metals Center, Tsukuba)
Keywords: *Java application, Windows software, scientific calculation, programming, personal computer*
2009年4月17日受理

単に作成できる利点があるが，セキュリティーの関係で，データのディスクへの保存機能がない．これでは数値計算してもデータを保存できないので，上記(3)の操作を実行できず，技術者・研究者における日常の数値計算の目的からはふさわしくないからである．ちなみに，Java アプリケーションを Java アプレットに書き直すことは，それほど難しくはなく，また Java アプレットに関しては多くの入門書が出版されているので，興味のある読者はトライされることを薦める．言語として Java を選んだ理由は，Java は OS に非依存であるので，MS-Windows でも MAC でも Linux でも，同一のソースプログラムにて計算が可能となるからである．特にグラフィックスに関するライブラリーが OS に依存しない点も重要である．さらにインターネットを通じて無料でプログラミング環境が入手できる点も重要であろう．

さて，3 回にわたる本講座にて説明するプログラムについては，以下を計画している．

第 1 回：自由エネルギーを例題とした数値計算，グラフの描画，およびデータの入出力

第 2 回：拡散相分離の Phase-field シミュレーションプログラム

第 3 回：無拡散変態の Phase-field シミュレーションプログラム

著者が相変態のシミュレーションを専門としているので，題材として，相変態・組織形成に関連する例を取り上げる．相変態・組織形成では，濃度プロファイル等のグラフだけでなく，2 次元や 3 次元計算における組織形態変化も計算対象とするので，画像・動画としての面白さも満喫していただけると嬉しい限りである．なお本稿では，Java 言語自体についての説明は省略させていただく．Java 全体に関しては文献(1)(2)，画像関連では文献(3)-(5)，工学的な数値計算については文献(6)-(9)などが参考になると思う．

1.2 Java 本体の入手先およびソースコードの実行方法

Java 本体のインストールの詳細については，Sun マイクロシステムズのホームページ(http://java.sun.com/javase/downloads/index.jsp)等を参照していただきたい．また以下，OS は MS-Windows XP 以降として話を進める．Java のバージョンに関しては，本計算では JDK 5 以降とし(執筆時における最新バージョンは，JDK 6 Update 13 である)，Java のインストール先については，上記 URL から Java 本体(JDK 6 Update 13 の場合には，jdk-6u13-windows-i586-p.exe)をダウンロードして，C ドライブの C:¥jdk1.6.0_13 にインストールしていただきたい(ディレクトリは新規に作成)．なお，Java の古いバージョンでは，環境変数 CLASSPATH の設定が必要な場合がある．これについても，本稿では具体的な説明はしないが，インターネットの探索サイトにおいて，キーワード(Java CLASSPATH)などで簡単に調べることができる(基本的にインターネットで簡単に検索できる内容については，本稿では説明を省略させていただく)．

さて，ソースコードのコンパイルおよび実行方法について説明しよう．ソースコードの名前をたとえば，“F_curve.java” とする．F_curve.java の形式は通常のテキストファイルである(MS-Windows の“メモ帳”などで読み書きできる形式)．ソースコードをコンパイルするために，バッチファイルを作成しておくと良い．著者は，**図 1** のようなバッチファイルを使用している．ファイル名は “vjc.bat” で(ファイル形式はテキストファイル)，内容を図 1 に示す(Java 本体は C:¥jdk1.6.0_13 にインストールされている場合を想定している)．このバッチファイル vjc.bat とソースコード F_curve.java は，C:¥tmp_program にあるものとする．コマンドプロンプト(通称 DOS 窓とも呼ばれ，MS-Windows のアクセサリー内にある．もし見当たらなければ，C:¥WINDOWS¥system32¥cmd.exe のショートカットを作成して，スクリーン上に持ってくると良い)を起動し，カレントディレクトリを C:¥tmp_program に移動して，C:¥tmp_program ＞vjc F_curve.java と入力しリターンすることによって，F_curve.java がコンパイルされる．エラーがなければクラスファイル F_curve.class が生成される．エラーがある場合には，エラーメッセージとそのエラーが存在するソースコード上の行番号がコマンドプロンプト画面に示されるので，エラーメッセージに従いソースコードを修正する．

プログラムの実行(得られた F_curve.class の実行)には，**図 2** のバッチファイルを使用すると良い(ファイル名を “vj.bat” とする)．具体的には，C:¥tmp_program ＞vj F_curve と入力(拡張子の class は必要ない)しリターンすることによって，プログラム(F_curve.class)を実行することができる(なおプログラムを停止するには，画面右上の×をクリックして，強制終了する)．上記のバッチファイルでは，毎回，現行の path を書き換えて，処理を行った後，再び path を元に戻す操作を行っており，あまりスマートではない．OS の path 設定にあらかじめ Java 関連の path を追加しておけば，この部分の操作は不要となる．

```
set JAVADir=c:¥jdk1.6.0_13
set path_1=%path%
path %JAVADir%¥bin;%JAVADir%¥include;%JAVADir%¥lib;
javac -deprecation %1 %2 %3 %4 %5 %6 %7 %8 %9
path ;
path %path_1%
```

図 1　vjc.bat の内容．

```
set JAVADir=c:¥jdk1.6.0_13
set path_1=%path%
path %JAVADir%¥bin;%JAVADir%¥include;
java %1 %2 %3 %4 %5 %6 %7 %8 %9
path ;
path %path_1%
```

図 2　vj.bat の内容．

1.3　プログラムの説明

以下では，実際の研究に役に立つ代表的な例題プログラム F_curve.java について説明する．F_curve.java は，以下の一連の操作を行っている．

(1)　濃度 c の自由エネルギー関数 G(c)を計算し，c と G(c)の離散数値データを配列に入力
(2)　c と G(c)の配列を描画(自由エネルギー曲線)
(3)　c と G(c)の配列をディスクに保存
(4)　(3)で保存された c と G(c)の配列を読み出し，別の配列に入力
(5)　(4)の配列データを再描画

したがって，データの数値計算，グラフ化，保存，および読出しの一連の操作が，このコードに含まれている．通常の技術系の計算では，以上が出来れば，ほとんどの作業が可能となる．なお自由エネルギーの数値データを計算したいだけならば，上記の(4)と(5)は必要ない．(4)と(5)はデータの読み出し部分の説明のために加えたものである．また上記の関数 G は，相互作用パラメータ 25 (kJ/mol)の正則溶体近似[10]の自由エネルギー曲線である．

さて，プログラム F_curve.java の全体構成は，**図 3** のようになっている．Java のプログラムの基本はクラス[1,2]であり，このプログラムは，F_curve という名前のクラスである(このプログラムには，グラフ表記が含まれるので，Frame クラスを継承して定義している)．図 3 の最初の import 文は C 言語における include 文に相当する．続いてクラスの中身について説明する．クラス全体で共通に使用する変数(や配列)が，「グローバル変数」である．コンストラクタは，このクラスの実体(この場合は画面)を作成する部分である(厳密ではないが，グラフを書くための下地の Window の設定と考えても良い)．メインプログラムが実際の計算処理および描画を行っているプログラム部分である．その後の 4 行がサブルーチンで，それぞれ，自由エネルギーの値を計算する部分，自由エネルギーのグラフの描画，数値データのハードディスクへの保存，およびハードディスクからの数値データの読み出しに対応している(プログラムの実行は，コマンドプロンプトから行うので，標準入出力画面はコマンドプロンプト画面となる)．

```
import文
public class F_curve extends Frame{
        グローバル変数
        public F_curve(){}                                  //コンストラクタ
        public static void main(String[] args){}            //メインプログラム
        double G(double c, double temp){}                   //自由エネルギー関数
        public void paint(Graphics g){}                     //自由エネルギーのグラフ描画
        private void datsave(){}                            //データの保存
        private void datin(){}                              //データの読込み
}
```

図 3　F_curve.java の構成．

さてソースコードの中身について具体的に見ていこう．Java では，ソースプログラムにおいて，ダブルスラッシュ“//”以降の 1 行がコメント文として扱われるので(C＋＋でも同様)，以下において，プログラムの内容を，ソースプログラム内にコメント文として，直接，書き込んで説明する．したがって，このソースコード(以下の文)は，そのままコンパイルすることができる．さて，説明に入ろう．以下，図 3 と対応させながら見て行っていただきたい．

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


public class F_curve extends Frame{

//***［グローバル変数］＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
      static int ND＝201;                      //濃度軸の分割数＋1
      static int ndm＝ND-1;                    //濃度軸の分割数
      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;              //円周率
      static double RR＝8.3145;                //ガス定数
      static double[]ch＝new double[ND];      //濃度
      static double[]Fh＝new double[ND];      //自由エネルギー
      static double[]c2h＝new double[ND];     //濃度(あらためてデータを読み出した時に使用)
      static double[]F2h＝new double[ND];
                                       //自由エネルギー(あらためてデータを読み出した時に使用)
      static int color_flg;                 //(ch,Fh)を描くか，(c2h,F2h)を描くかを区別するフラグ
      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

//***［コンストラクタ］＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
      public F_curve(){
        xwidth＝600; 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 をセット
        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 を閉じる時の設定(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

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

        int i;             //整数
        double temp, c;    //温度，組成

//----  温度を設定-----------------------------------------------------------------------------------
        temp＝1000.0;      //[K]

//----  標準入出力(コマンドプロンプト)より，温度を入力する場合------------------------------------------
//      Java の場合，標準入出力からの数値データ入力は，下記のように面倒であるので，
//      ここでは，参考までに定番の書式を示しておく
//
//        String s_temp;       //温度(文字列)
//        BufferedReader input＝new BufferedReader(new InputStreamReader(System.in));
//        s_temp＝''1000.0''; //標準値を設定
//        try{System.out.print(''temp(1000.0)/K＝  '');s_temp＝input.readLine();}
//        catch(IOException e){System.out.println(''Exception : ''＋e);}
//        temp＝new Double(s_temp).doubleValue();       //文字列を数値に変換

//---   自由エネルギーの計算----------------------------------------------------------------------------
        for(i＝0; i＜＝ndm; i＋＋){
          c＝ch[i]＝(double)i/(double)ndm;
          if(c＝＝0.0){c＝1.0e－6;}if(c＝＝1.0){c＝1.0－1.0e－6;}
                                                    //c＝1 と c＝0 の log 関数エラーの回避
          Fh[i]＝prog.G(c,temp);                   //自由エネルギーのサブルーチンを呼んでいる
          System.out.printf(''%f  %e  ¥n'',c,Fh[i]);
                                                 //標準入出力へ，濃度と自由エネルギー値を表示
        }

//---   自由エネルギーの描画----------------------------------------------------------
        color_flg＝0; prog.repaint();

//---   データの保存------------------------------------------------------------------
        prog.datsave();

//---   描画が速すぎるので，強制的に 5 秒スリープ-------------------------
        Thread.sleep(5000);
                                                                                          ↗
```

```
//---  上記で保存したデータを別の配列に読み込み--------------------------
       prog.datin();

//---  新しく読み込んだ配列の自由エネルギーを再描画--------------------
       color_flg＝1; prog.repaint();
//-------------------------------------------------------------------------------------

  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
//***［自由エネルギー関数(A-B2 成分系の正則溶体近似)］＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
     double G(double c, double temp){
       double c1, c2;
       double L0, gc;

       if(c＝＝0.0){c＝1.0e－10;}  if(c＝＝1.0){c＝1.0－1.0e－10;}
                                                  //c＝1 と c＝0 の log 関数エラーの回避
       c1＝1.0－c; c2＝c;                          //組成
       L0＝2.5e＋04;                               //相互作用パラメータ 25[kJ/mol]
       gc＝L0*c1*c2＋RR*temp*(c1*Math.log(c1)＋c2*Math.log(c2)); //自由エネルギー式
       return(gc);
     }

//***［自由エネルギーのグラフ描画］＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
     public void paint(Graphics g){

       g.clearRect(0, 0, width, height);  //Window をクリア
       int i, i1, i2;  //整数
       double xmax, xmin, dx, ymax, ymin, dy;
       double gx1, gy1, gx2, gy2;
       int ixmax, iymax, ixmin, iymin, igx1, igy1, igx2, igy2;
       int idx, idy, ir;

       xmin＝0.0; xmax＝1.0; dx＝0.1;          //横軸の最小値，最大値，分割間隔(実際の値)
       ymin＝－1000.0; ymax＝1000.0; dy＝100.0;//縦軸の最小値，最大値，分割間隔(実際の値)
       ixmin＝0; iymin＝0; ixmax＝xwidth; iymax＝yheight;  //ピクセル単位の場合
       ir＝4;                                 //青丸の半径

       idx＝(int)(ixmax*(dx/(xmax－xmin))＋0.5);  //ピクセル単位における横方向の分割間隔
       idy＝(int)(iymax*(dy/(ymax－ymin))＋0.5);  //ピクセル単位における縦方向の分割間隔

       g.setColor(Color.white);                   //色を白に指定
       g.fillRect(insetx, insety, xwidth, yheight);  //画面を上で指定した色で塗る

       g.setColor(Color.lightGray);               //色を灰色に指定
       g.drawRect(insetx, insety, xwidth, yheight);  //グラフの外周を上で指定した色で描く
       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);}
                                                  //グラフ内の縦横の線を等間隔に描く
//-------------------------------------------------------------------------------
       if(color_flg＝＝0){  //最初の描画(color_flg＝0の場合)
         g.setColor(Color.red);                   //色を赤に指定
         for(i＝0; i＜ndm; i＋＋){
           i1＝i; i2＝i＋1;                       //個々の隣り合う 2 点を直線で連続的に結ぶ
           gx1＝ch[i1]; gy1＝Fh[i1]; gx2＝ch[i2]; gy2＝Fh[i2];  //濃度と自由エネルギーの値

           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));
                              //濃度と自由エネルギーの値をスクリーン上のピクセル値に変換
           g.drawLine(insetx＋igx1, insety＋igy1, insetx＋igx2, insety＋igy2);
                                                          //2 点を直線で結ぶ
         }
       }
       else{//再描画の場合(color_flg＝1 の場合)
         g.setColor(Color.blue);                  //色を青に指定
         for(i＝0; i＜＝ndm; i＋＋){              //個々の点に小さな青丸を描く
           gx1＝c2h[i]; gy1＝F2h[i];              //濃度と自由エネルギーの値
           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));
                              //濃度と自由エネルギーの値をスクリーン上のピクセル値に変換
           g.fillOval(insetx＋igx1－ir, insety＋igy1－ir, 2*ir, 2*ir);        //青丸の描画
         }
       }
     }

//***［データの保存］＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
     private void datsave()throws Exception{
       int i;
       //保存先ファイル名を ini000.dat とする
       PrintWriter outfile＝new PrintWriter
                                  (new BufferedWriter(new FileWriter(''ini000.dat'')));
                                                  //ファイルのオープン(上書きの場合)
       //PrintWriter outfile＝new PrintWriter(new BufferedWriter
       //                                  (new FileWriter(''ini000.dat'', true)));
                                                  //ファイルのオープン(追記の場合)
       for(i＝0; i＜＝ndm; i＋＋){
         outfile.println(ch[i]);  //データの書き込み
         outfile.println(Fh[i]);  //データの書き込み
       }
       outfile.close();           //ファイルのクローズ
     }

//***［データの読み出し］＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
     private void datin()throws Exception{
       int i;
       String s_data;

       //入力ファイル名を ini000.dat とする
       BufferedReader infile＝new BufferedReader(new FileReader(''ini000.dat'');
                                                    //ファイルのオープン
         for(i＝0; i＜＝ndm; i＋＋){
           s_data＝infile.readLine();               //文字列として読み込み
           c2h[i]＝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
           F2h[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
     }

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

図 4 は上記プログラムを実行した時の自由エネルギー曲線である．縦軸が自由エネルギーで，横軸が組成，始めに図 4(a)の曲線が表示され，5 秒ほどしてから，図 4(b)のように曲線が上書きされる．ここでのグラフ表示は，計算時の結果確認を意図しており，キレイなグラフ作成を目的としていない．したがって，単に曲線の描画のみにとどめ，縦軸や横軸の説明などの表示は全て省略している．曲線の数値データはハードディスクに保存されているので，たとえば論文用などの図を描きたい時には，この数値データを各種のグラフ作成ソフト等に読み込んで綺麗に清書すればよい．最近では，非常に優れたグラフ作成ソフトや可視化ソフトが簡単に入手できるようになった．多くのグラフ作成ソフトには，関数を定義して曲線を描く機能は内蔵されている．関数が初等的で簡単な式ならば問題はないが，数値積分や収束計算などが含まれる場合には役にたたない．ここで説明した手法ならば，もともとソースコードを自分で書いているのであるから，いかなる計算にも対処でき，ソフトの制約を受けない．

1.4 おわりに

F_curve.java は 1 例題に過ぎないが，これを修正・改良することによって，様々な技術系の数値計算に対応したプログラムを自由に作成することができる．実際，次回以降の組織形成シミュレーションでは，F_curve.java を修正してプログラミングしている．また本プログラムは各種の実験データの解析などにも役立つ．筆者の経験から，Java は勉強するのに少しバリヤーがあると思う．本稿でそのバリヤーを幾分でも下げることができたならば望外の喜びである．特に本稿を読まれた学生さんに申し上げたい．本コードを書き直して自在に使いこなせるようになったならば，これは間違いなく，あなたの一生の財産になります．

なお本入門講座にて説明するソースコードやバッチファイルは，全て著者のホームページ(http://www.nims.go.jp/mpsg/Phase-Field_Modeling.htm)よりダウンロードできるので，ご自由にお使いいただきたい(ただし本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

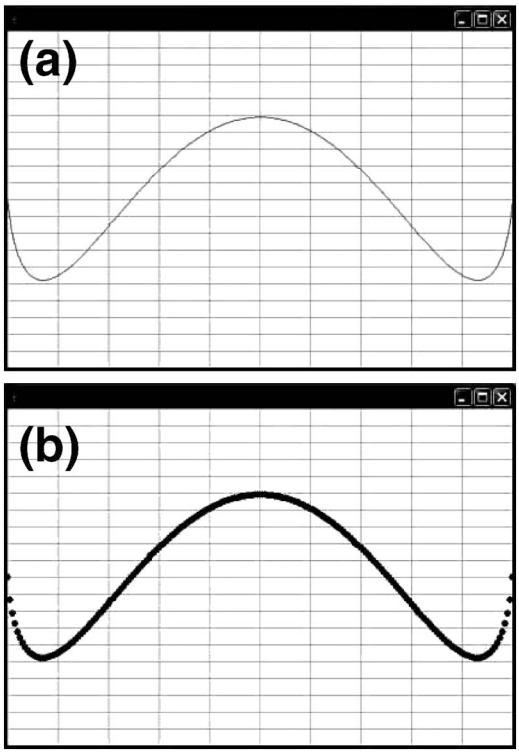

図 4　自由エネルギー曲線の計算結果.
(a) 1 回目の描画，(b) 2 回目の描画.

日本金属学会は責任を負いませんのでご了承ください).

次回は，拡散相分離における 1 次元濃度プロファイルの時間発展，および 2 次元濃度場の時間発展のプログラムについて解説する．また今回省略した画像関連の事項についても若干説明する予定である.

文　　献

(1) 宮坂雅輝：エッセンシャル Java(2nd edition)，ソフトバンククリエイティブ，(2003).
(2) 桑原恒夫：3 日で解る Java―例題学習方式(第 2 版版)，共立出版，(2000).
(3) 中山　茂：Java2 グラフィックスプログラミング入門，技報堂出版，(1999).
(4) 山本芳人：Java による図形処理入門，工学図書，(1998).
(5) 赤間世紀：Java による画像処理プログラミング，工学社，(2007).
(6) 峯村吉泰：Java で学ぶシミュレーションの基礎，森北出版，(2006).
(7) 峯村吉泰：Java によるコンピュータグラフィックス―基礎からシミュレーションの可視化まで，森北出版，(2003).
(8) 峯村吉泰：Java による流体・熱流動の数値シミュレーション，森北出版，(2001).
(9) 矢部　孝，尾形陽一，滝沢研二：CIP 法と Java による CG シミュレーション，森北出版，(2007).
(10) たとえば，阿部秀夫：金属組織学序論，コロナ社，(1967).

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.
★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★