2007 年度 卒業研究レポート

１・２次元熱伝導・波動方程式の動画化

明治大学 理工学部 数学科
158R038007 奥山恵史

2008 年 3 月 5 日

目 次

第1章　初めに

1.1　概要

2001年度卒業生の三井康之さんの「Java による波動方程式の数値解析」[1] の中のプログラムを、入手してコンパイル・実行せずとも見られる様に、Mpeg形式の動画ファイルにすることを目標としました。

なぜ動画ファイルを作るのかは、複雑な計算の必要なプログラムになってしまうと計算時間も膨大となり、1回グラフを描写するのにかなり時間がかかる様になってしまうので、あらかじめ計算をPCにさせて動画ファイルにしておけば、時間がかからずにシミュレーション結果を見られる様にするためです。

そのためにプログラムによって描かれるグラフを画像ファイルのJpeg形式に保存し、保存したJpegファイルを1つのMpeg形式の動画に変換することにより出来ました。

出来上がったMpegファイルをまとめたWWWページとして、`http://www.math.meiji.ac.jp/~mk/labo/2007/okuyama/index.html` があります。

第2章　波動方程式の差分法

2.1　問題

波動方程式

$$\frac{1}{c^2}u_{tt} = u_{xx} + u_{yy} \ \ (t > 0, (x, y) \in \Omega) \ , \ \Omega = \{(x, y); a < x < b, c < y < d\}$$

と初期条件　$u(x, y, 0) = \phi(x, y), u_t(x, y, 0) = \psi(x, y) \ \ ((x, y) \in \bar{\Omega})$
と次の境界条件のいずれか
(Dirichlet 境界条件)$u(x, y, t)|_{\partial\Omega} = 0 \ (t > 0)$
(Neumann 境界条件) $\frac{\partial u}{\partial n}(x, y, t)\big|_{\partial\Omega} = 0 \ (t > 0)$
からなる初期値境界値問題の差分方程式を求める。

2.2　差分方程式の導出

領域 Ω を座標軸に平行な格子点で x 軸、y 軸方向にそれぞれ N_x,N_y 等分し、格子点をそれぞれ x_i, y_j と表す。すなわち

$$d_x = \frac{b-a}{N_x} \ , \ \ d_y = \frac{d-c}{N_y}$$

として

$$x_i = a + id_x \ , \ \ y_j = c + jd_y \ \ (i = 0, 1, 2, \ \dots \ , N_x \ , \ j = 0, 1, 2, \ \dots \ , N_y)$$

とおく。

また $\tau > 0$ を固定して

$$t_k = k\tau \ \ (k = 0, 1, 2, \ \dots)$$

とおく。格子点 (x_i, y_j, t_k) において偏導関数を 2 階中心差分近似をして求めると

$$u_{tt} = \frac{u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}}{\tau^2} + O(\tau^2) \ ,$$

$$u_{xx} = \frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{d_x^2} + O(d_x^2) \ ,$$

$$u_{yy} = \frac{u_{i,j+1,} - 2u_{i,j,k} + u_{i,j-1,k}}{d_y^2} + O(d_y^2) \ .$$

ただし、$u_{i,j,k} = u(x_i, y_j, t_k)$ とした。この波動方程式に対する差分方程式として

$$\frac{1}{c^2}\frac{U_{i,j}^{k+1} - 2U_{i,j}^k + U_{i,j}^{k-1}}{\tau^2} = \frac{U_{i+1,j}^k - 2U_{i,j}^k + U_{i-1,j}^k}{d_x^2} + \frac{U_{i,j+1}^k - 2U_{i,j}^k + U_{i,j-1}^k}{d_y^2}$$

が得られる。$\lambda_x = \dfrac{c\tau}{d_x}, \lambda_y = \dfrac{c\tau}{d_y}$ とおいて、両辺に τ^2 をかけて移項をすると

$$U_{i,j}^{k+1} = 2(1-\lambda_x^2-\lambda_y^2)U_{i,j}^k + \lambda_x^2(U_{i+1,j}^k + U_{i-1,j}^k) + \lambda_y^2(U_{i,j+1}^k + U_{i,j-1}^k) - U_{i,j}^{k-1} \ .$$

$u(x_i, y_j, t)$ を $t=0$ を中心として Taylor 展開すると

$$\begin{aligned} u(x_i, y_j, t_1) &= u(x_i, y_j, 0) + u_t(x_i, y_j, 0)\tau + \frac{u_{tt}(x_i, y_j, 0)}{2!}\tau^2 + O(\tau^3) \\ &= u(x_i, y_j, 0) + \psi(x_i, y_j)\tau + \frac{u_{tt}(x_i, y_j, 0)}{2!}\tau^2 + O(\tau^3) \ . \end{aligned}$$

$t=0$ でも波動方程式が成り立つとすると

$$\begin{aligned} u(x_i, y_j, t_1) &= u(x_i, y_j, 0) + \psi(x_i, y_j)\tau + c^2(\frac{u_{xx}(x_i, y_j, 0)}{2!} + \frac{u_{yy}(x_i, y_j, 0)}{2!})\tau^2 + O(\tau^3) \\ &= u(x_i, y_j, 0) + \psi(x_i, y_j)\tau \\ &\quad + \frac{c^2}{2!}\left(\frac{u_{i+1,j,0} - 2u_{i,j,0} + u_{i-1,j,0}}{d_x^2} + \frac{u_{i,j+1,0} - 2u_{i,j,0} + u_{i,j-1,0}}{d_y^2}\right)\tau^2 \\ &\quad + O(d_x^2) + O(d_y^2) + O(\tau^3) \ . \end{aligned}$$

これから次の様な差分方程式が得られる。

$$U_{i,j}^1 = U_{i,j}^0 + \psi(x_i, y_j)\tau + \frac{c^2}{2!}\frac{U_{i+1,j}^0 - 2U_{i,j}^0 + U_{i-1,j}^0}{d_x^2}\tau^2 + \frac{c^2}{2!}\frac{U_{i,j+1}^0 - 2U_{i,j}^0 + U_{i,j-1}^0}{d_y^2}\tau^2.$$

整理すると

$$U_{i,j}^1 = (1-\lambda_x^2-\lambda_y^2)U_{i,j}^0 + \psi(x_i, y_j)\tau + \frac{\lambda_x^2}{2}(U_{i+1,j}^0 + U_{i-1,j}^0) + \frac{\lambda_y^2}{2}(U_{i,j+1}^0 + U_{i,j-1}^0).$$

2.3 差分方程式

以上をまとめると

$$\begin{aligned} &U_{i,j}^{k+1} = 2(1-\lambda_x^2-\lambda_y^2)U_{i,j}^k + \lambda_x^2(U_{i+1,j}^k + U_{i-1,j}^k) + \lambda_y^2(U_{i,j+1}^k + U_{i,j-1}^k) - U_{i,j}^{k-1} \\ &\qquad (i = 1, 2, \ \dots \ , N_x - 1; j = 1, 2, \ \dots \ , N_y - 1; k = 1, 2, \ \dots) \ , \\ &U_{i,j}^0 = \phi(x_i, y_i) \quad (0 \le i \le N_x \ , \ 0 \le j \le N_y) \ , \\ &U_{i,j}^1 = (1-\lambda_x^2-\lambda_y^2)U_{i,j}^0 + \psi(x_i, y_j)\tau + \frac{\lambda_x^2}{2}(U_{i+1,j}^0 + U_{i-1,j}^0) + \frac{\lambda_y^2}{2}(U_{i,j+1}^0 + U_{i,j-1}^0) \\ &\qquad (1 \le i \le N_x - 1 \ , \ 1 \le j \le N_y - 1) \ . \end{aligned}$$

また Dirichlet 境界条件は

$$U_{0,j}^k = U_{N_x,j}^k = 0 \ , \ U_{i,0}^k = U_{i,N_y}^k = 0 \ \ (i = 0, 1, 2, \ \dots \ , N_x \ , \ j = 0, 1, 2, \ \dots \ , N_y \ , \ k = 1, 2, \ \dots)$$

Neumann 境界条件は

$$U^k_{1,j} = U^k_{0,j}\ ,\ U^k_{N_x,j} = U^k_{N_x-1,j}\ \ ,\ \ U^k_{i,1} = U^k_{i,0}\ ,\ U^k_{i,N_y} = U^k_{i,N_y-1}$$
$$(i = 0, 1, 2,\ \ldots\ , N_x\ ,\ j = 0, 1, 2,\ \ldots\ , N_y\ ,\ k = 1, 2,\ \ldots)$$

となる。

2.4 プログラム

2.3 節で求まった差分解を使っているのは付録 A.1 の wave2d_s.java のプログラムです。

初期値を

$$\phi(x,y) = \begin{cases} \dfrac{\sin\{(2x-0.5)\pi\}+1}{4} & (-2.0 \le x \le -1.0) \\ \\ 0 & (-1.0 < x \le 2.0), \end{cases}$$

$$\psi(x,y) = \begin{cases} \dfrac{-2\pi\cos\{(2x-0.5)\pi\}}{4} & (-2.0 \le x \le -1.0) \\ \\ 0 & (-1.0 < x \le 2.0) \end{cases}$$

としています。

第3章　Jpeg形式に保存、Mpeg形式に

3.1　Javaプログラムから Jpeg ファイルの作成の仕方

付録 A.1 のプログラム wave2d_s.java(と A.3 の主要部) を説明していきます。

```
BufferedImage im= new BufferedImage(ImgX, ImgY, BufferedImage.TYPE_3BYTE_BGR);}
```

でファイル出力するためのバッファーを備えた Image を確保し (A.1 では ℓ25 と ℓ58、A.3 では ℓ16)、

```
Graphics bg = im.getGraphics();
```

でその Graphics を取得しています (A.1 では ℓ28 と ℓ59、A.3 では ℓ17)。

```
MitsuiWorld_2 m = new MitsuiWorld_2();
```

で m.〜という関数を使える様にし、(A.1 では ℓ19、A.3 では ℓ42)

```
m.setGraphics(bg);
```

で bg と m のつながりを持たせることで (A.1 では ℓ106、A.3 では ℓ44)、m を通しての描画、例えば

```
m.hideBirdView(u, nx, ny, 1);
```

では im になされる様になります (A.1 では ℓ94 で関数の宣言し、ℓ155 と ℓ180 で呼び出し、A.3 では m.draw というのが最初からあるので、使っていません)。

```
public void paint(Graphics g)  {g.drawImage(im, 0, 0, this);}
```

で画面に表示されます (A.1 では ℓ98 で関数の宣言し、ℓ156 と ℓ181 で呼び出し、A.3 では ℓ64)。

```
ImageIO.write(im, "jpeg", new File("test" + str + ".jpg")
```

で Jpeg ファイルに出力しています (A.1 では ℓ124 で関数の宣言し、ℓ157 と ℓ182 で呼び出し、A.3 では ℓ21 で関数の宣言し、ℓ65 で呼び出し)。
Jpeg ファイルは img0000.jpg〜img9999.jpg の様にファイル名を連番にしておくと使おうとしている TMPGEnc のソフトで Mpeg ファイルに変換できます。

3.2 MitsuiWorldのプログラムへの組み込み例１

付録 A.2 の Ex1.java というプログラムは、MitsuiWorld のパッケージを使った簡単なプログラムで 1 次元の $\sin(\pi x)$ のグラフを描くものです。

このプログラムを修正して描いたグラフを画像ファイルとして記録出来る様にしたプログラムが付録 `A.3` の `Ex1_new_ver_1.java` です。

変更部分は主に//add とある部分です。

- ℓ1〜ℓ5 の注釈部分の Applet code の部分は必要ないので削除しています。
- ℓ6 の部分は Applet のものを使わないので削ることも出来ます。
- ℓ9,ℓ11,ℓ12 の部分は ImageIO.write の関数を使うための宣言です。
- ℓ10 の部分は ℓ16 の BufferedImage の関数を使うための宣言です。
- ℓ14 では Applet のままだとファイルへの書き込みが出来ないために Frame にしています。
- ℓ18 では、ImageIO.write の関数を使うために result という変数を宣言しています。
- ℓ28 は Frame にしたために必要になった部分です。
- ℓ29 で画面に表示させるウィンドウのサイズを指定し、
- ℓ30〜ℓ34 はウィンドウをウィンドウ右上の×ボタンで消せる様にするための記述です。
- ℓ35 はレイアウトを自由に出来るための宣言です。
- ℓ36 で内容を表示出来る様にしています。
- ℓ38〜ℓ40 も Frame にしたために必要になった部分です。
- ℓ47〜ℓ49 で画面を一度真っ白にすることによりグラフが描ける様にしています。

3.3 MitsuiWorld のプログラムへの組み込み例２

付録 A.4 の wave2d_simple_1.java は２次元波動方程式を時間ごとの変化が見られる様になっているプログラムです。このプログラムで描いたグラフを画像ファイルに記録する様に修正したものが、付録 A.5 の wave2d_simple_new_ver_1.java のプログラムです。

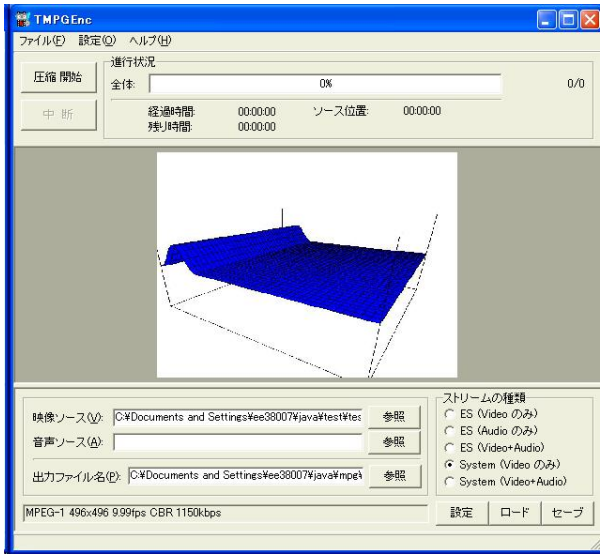
A.2 → A.3 の変え方と違う部分は、

- ℓ23、ℓ25〜ℓ35、ℓ38 を加え、ℓ37 を変えることにより、
  3.1 節の最後の部分で述べた様に、TMPGEnc のソフトを使うために Jpeg ファイルを img0000.jpg 〜img9999.jpg の様にファイル名を連番にする様にしています。
- ℓ54 を加えることにより、A.2 に無かった init() に対応させています。
- ℓ60 は ℓ21 を加えているので、必要でなくなったために削除です。
- ℓ105 は機能しなくなったために削除で、代わりに ℓ105〜ℓ108 で済ませます。
- ℓ129〜ℓ132 も同様です。

3.4 Jpeg ファイルの Mpeg ファイルへの変換の仕方

- TMPGEnc というソフトウェアの
  フリーで使える部分のみを使い作成
- 操作の方法は
  1. 映像ソースの初めの Jpeg ファイルを選択
  2. 出力ファイル名を決め
  3. ストリームの種類を System(Video のみ)
  4. 圧縮開始

これで、Mpeg ファイルを作ることが出来ます。



TMPGEnc のソフトは株式会社ペガシス社のホームページの `http://www.tmpgenc.net/ja/j_download.html` から TMPGEnc 無料版がダウンロード出来ます。
ダウンロードして使おうとして TMPGEnc.exe を起動すると右図の様な画面の上から重なりプロジェクトウィザードという画面が出ますが、これは関係ないのでキャンセルのボタンをクリックします。
Mpeg の再生時間を調整したい場合は右下の設定ボタンをクリックして、ビデオタブのフレームレートのところを、再生時間を短くしたければ値を大きくし、再生時間を長くしたければ値を小さくすると出来ます。

付 録A　ソースプログラム

A.1　wave2d_s.java

2 次元波動方程式を描き、描いたグラフを画像ファイルの Jpeg 形式に記録出来る様にしたプログラムです。
波動方程式

$$\frac{1}{c^2}u_{tt} = u_{xx} + u_{yy} \ \ (t > 0, (x, y) \in \Omega) \ , \ \Omega = \{(x, y); -2 < x < 2, 2 < y < 2\}$$

と初期条件　$u(x, y, 0) = \phi(x, y), u_t(x, y, 0) = \psi(x, y) \ \ ((x, y) \in \bar{\Omega})$
と境界条件
(Dirichlet 境界条件)$u(x, y, t)|_{\Gamma_1} = 0$ $(\Gamma_1 = \{-2 \leq x \leq 2 \ , \ y = -2, 2\} \ , \ t > 0)$
(Neumann 境界条件) $\frac{\partial u}{\partial n}(x, y, t)\Big|_{\Gamma_2} = 0$ $(\Gamma_2 = \{x = -2, 2 \ , \ -2 \leq y \leq 2\} \ , \ t > 0)$
からなる初期値境界値問題です。
初期値を

$$\phi(x, y) = \begin{cases} \dfrac{\sin\{(2x - 0.5)\pi\} + 1}{4} & (-2.0 \leq x \leq -1.0) \\ \\ 0 & (-1.0 < x \leq 2.0) \end{cases}$$

$$\psi(x, y) = \begin{cases} \dfrac{-2\pi\cos\{(2x - 0.5)\pi\}}{4} & (-2.0 \leq x \leq -1.0) \\ \\ 0 & (-1.0 < x \leq 2.0) \end{cases}$$

としています。
次の様にしてコンパイルします。

```
javac wave2d_s.java
```

実行は

```
java wave2d_s
```

```
1   /*
2    * wave2d_s.java --- 2 次元波動方程式
3    *   コンパイルは javac wave2d_s.java
4    *   実行は java wave2d_s
5    * MistuiWorld を使い、2 次元波動方程式をに出力する。
6    * 4-1 目:平面波を x 軸 Dirichlet 境界条件、y 軸 Neumann 境界条件に。
7    * 1 コマずつ消去。
8    * 変数入力を簡素に。
9    */
10
```

```
11  import java.awt.*;
12  import java.awt.event.*;
13  import java.awt.image.BufferedImage;
14  import java.io.File;
15  import javax.imageio.ImageIO;
16  import Mitsui.*;
17
18  public class wave2d_s extends Frame implements Runnable{
19      MitsuiWorld_2 m = new MitsuiWorld_2();
20      private static final long serialVersionUID = 1L;
21
22      static final int ImgX = 496, ImgY = 496;
23      static final int WinX = ImgX, WinY = ImgY;
24      Thread th = null;
25      BufferedImage im = null;
26      Graphics bg = null;
27      int nx = 40;
28      int ny = 40;
29      double tau = 0.05;
30      double Tmax = 8.8;
31      double xmin = -2.0;
32      double xmax = 2.0;
33      double ymin = -2.0;
34      double ymax = 2.0;
35      double zmin = -1.0;
36      double zmax = 1.0;
37      double dx = (xmax-xmin)/nx;
38      double dy = (ymax-ymin)/ny;
39      double lambdax = tau / dx;
40      double lambday = tau / dy;
41      double lambdax2 = lambdax * lambdax;
42      double lambday2 = lambday * lambday;
43      double[][] u1, u2, u3;
44      int kmax = (int)Math.rint(Tmax/ tau);
45      private boolean result = false;
46      private int jpeg_file_number = 0;
47      public wave2d_s() {
48          this.setSize(WinX, WinY);
49          this.addWindowListener(new WindowAdapter() {
50              public void windowClosing(WindowEvent ev) {
51                  System.exit(0);
52              }
53          });
54          setLayout(null);
55          this.setVisible(true);
56          // ダブル・バッファリング用のバッファーを準備
57          if (im == null) {
58              im = new BufferedImage(ImgX, ImgY, BufferedImage.TYPE_3BYTE_BGR);
59              bg = im.getGraphics();
60          }
61          // 計算開始の準備
```

```
62          initcomputation();
63          // 計算スレッドを開始する
64          this.start();
65      }
66      public void start() {
67          if (th == null) {
68              th = new Thread(this);
69              th.start();
70          }
71      }
72      public void stop() {
73          if (th != null) {
74              th = null;
75          }
76      }
77      public static void main(String[] args) {
78          new wave2d_s();
79      }
80      // 初期データ
81      public double phi(double x,double y){
82          if(xmin<=x && x<=xmin+1)
83                  return (Math.sin(Math.PI*(x*2-0.5))+1)/4;
84              else
85                  return 0.0;
86       }
87      public double psi(double x,double y){
88          if(xmin<=x && x<=xmin+1)
89                  return -2*Math.PI*Math.cos(Math.PI*(x*2-0.5))/4;
90              else
91                  return 0.0;
92       }
93      // グラフ化
94      private void drawGraph(double[][] u) {
95          m.hideBirdView(u, nx, ny, 1);
96      }
97      // ダブルバッファリングの定跡
98      public void paint(Graphics g) {
99          g.drawImage(im, 0, 0, this);
100     }
101     // 計算の準備 (変数の準備)
102     private void initcomputation() {
103         u1 = new double[nx+1][ny+1];
104         u2 = new double[nx+1][ny+1];
105         u3 = new double[nx+1][ny+1];
106         m.setGraphics(bg);
107         m.setScreenSize(getSize().width, getSize().height);
108         m.setColor(2);
109     }
110     // 現在のバッファー im の内容を JPEG 形式で書き出す
111     void saveJpeg() {
112         String str;
```

```
113             if (jpeg_file_number < 10)
114                 str = "0000" + jpeg_file_number;
115             else if (jpeg_file_number < 100)
116                 str = "000" + jpeg_file_number;
117             else if (jpeg_file_number < 1000)
118                 str = "00" + jpeg_file_number;
119             else if (jpeg_file_number < 10000)
120                 str = "0" + jpeg_file_number;
121             else
122                 str = "" + jpeg_file_number;
123             try {
124                   result = ImageIO.write(im, "jpeg", new File("test" + str + ".jpg"));
125                   jpeg_file_number++;
126                 }
127                 catch (Exception e) {
128                   e.printStackTrace();
129                   result = false;
130                 }
131         }
132         // 計算スレッド
133         public void run() {
134                         for(int i=0; i<=nx; i++)
135                            for(int j=0; j<=ny; j++)
136                               u1[i][j] = phi(xmin+i*dx,ymin+j*dy);
137                         for(int i=1; i<nx; i++)
138                            for(int j=1; j<ny; j++)
139                               u2[i][j] = (1.0-lambdax2-lambday2) * u1[i][j] +
140                                  0.5 * lambdax2 * (u1[i-1][j]+u1[i+1][j]) +
141                                  0.5 * lambday2 * (u1[i][j-1]+u1[i][j+1]) +
142                                  tau * psi(xmin+i*dx,ymin+j*dy);
143                         //x 軸 Neumann 境界条件、y 軸 Dirichlet 境界条件
144                         for(int i=0;i<=nx;i++){
145                             u2[i][0]=u2[i][1];
146                             u2[i][ny]=u2[i][ny-1];
147                         }
148                         for(int i=0;i<=ny;i++)
149                             u2[0][i]=u2[nx][i]=0.0;
150                       bg.setColor(Color.white);
151                       bg.fillRect(0, 0, ImgX, ImgY);
152                       bg.setColor(Color.black);
153 
154                       m.setArea2(xmin, xmax, ymin, ymax, zmin, zmax);
155                       drawGraph(u2);
156                       repaint();
157                       saveJpeg();
158                       try {
159                           Thread.sleep(300);
160                       } catch (Exception ex) {
161                           ex.printStackTrace();
162                       }
163 
```

```
164                     for(int k= 1; k<= kmax; k++){
165                       for(int i=1;i<nx;i++)
166                         for(int j=1;j<ny;j++)
167                           u3[i][j] = 2 * (1.0-lambdax2 -lambday2) * u2[i][j] +
168                             lambdax2 * (u2[i-1][j]+u2[i+1][j]) +
169                             lambday2 * (u2[i][j-1]+u2[i][j+1]) - u1[i][j];
170                       //x 軸 Neumann 境界条件、y 軸 Dirichlet 境界条件
171                       for(int i=0;i<=nx;i++){
172                           u3[i][0] = u3[i][1] ;
173                           u3[i][ny]= u3[i][ny-1];
174                       }
175                       for(int i=0;i<=ny;i++)
176                           u3[0][i]=u3[nx][i]=0;
177                       bg.setColor(Color.white);
178                       bg.fillRect(0, 0, ImgX, ImgY);
179                       m.setArea2(xmin, xmax, ymin, ymax, zmin, zmax);
180                       drawGraph(u3);
181                       repaint();
182                       saveJpeg();
183                       for(int i=0;i<=nx;i++){
184                          for(int j=0;j<=ny;j++){
185                              u1[i][j] = u2[i][j];
186                              u2[i][j] = u3[i][j];
187                          }
188                       }
189                       try {
190                         Thread.sleep(10);
191                       } catch (Exception ex) {
192                           ex.printStackTrace();
193                       }
194                     }
195         }
196     }
```

A.2 Ex1.java

1 変数関数

$$f(x) = \sin \pi x$$

のグラフを描くプログラムです。

次の様にしてコンパイルします。

```
javac Ex1.java
```

実行は

```
appletviewer Ex1.java &
```

```
1   /*
2    * Ex1.java
```

```
 3    *   <applet code="Ex1.class" width=500 height=500></applet>
 4    */
 5   import java.applet.Applet;
 6   import java.awt.*;
 7   import Mitsui.*;                                    //パッケージを呼び込む
 8
 9   public class Ex1 extends Applet{
10       public void paint(Graphics g){                      //applet の paint 部分
11           MitsuiWorld_2 m = new MitsuiWorld_2();              //インスタンス生成
12           m.setGraphics(g);
13           m.setScreenSize(getSize().width,getSize().height);   //描く画面をセット
14           // [a, b] × [c, d]
15           m.setArea(-1.0,1.0,-1.0,1.0);                       //描写範囲を決める
16
17           //座標軸を設定
18           m.move(-1.0,0.0); m.draw(1.0,0.0);
19           m.move(0.0,1.0); m.draw(0.0,-1.0);
20
21           m.setColor(Color.red);                          //色を赤にする
22
23           double h=0.01;                                 //sin(π x) を 0.01 刻みで描く
24           double x=-1.0;
25           m.move(x,f(x));
26           for( x=-1.01;x<=1.0;x+=h){
27               m.draw(x,f(x));
28           }
29       }
30
31       public double f(double x){
32           return Math.sin(Math.PI*x);
33       }
34   }
```

A.3 Ex1_new_ver_1.java

Ex1.java のプログラムを修正し、描いたグラフを画像ファイルの Jpeg 形式に記録出来る様にしたプログラムです。
次の様にしてコンパイルします。

```
javac Ex1_new_ver_1.java
```

実行は

```
java Ex1_new_ver_1
```

```
1   /*
2    * Ex1_new_ver_1.java
3    * コンパイルは javac Ex1_new_ver_1.java
4    * 実行は java Ex1_new_ver_1
5    */
```

```
6  //import java.applet.Applet;
7  import java.awt.*;
8  import Mitsui.*;                                    //パッケージを呼び込む
9  import java.awt.event.*;              //add
10 import java.awt.image.BufferedImage;  //add
11 import java.io.File;                  //add
12 import javax.imageio.ImageIO;         //add
13
14 public class Ex1_new_ver_1 extends Frame {             //Applet → Frame
15
16    BufferedImage im = new BufferedImage(500, 500, BufferedImage.TYPE_3BYTE_BGR); //add
17    Graphics bg = im.getGraphics();                     //add
18    private boolean result = false;                     //add
19    void saveJpeg() {                                   //add
20       try {
21           result = ImageIO.write(im, "jpeg", new File("a.jpg"));
22       }
23       catch (Exception e) {
24          e.printStackTrace();
25          result = false;
26       }
27    }
28    public Ex1_new_ver_1() {                              //add
29       this.setSize(500, 500);
30       this.addWindowListener(new WindowAdapter() {
31          public void windowClosing(WindowEvent ev) {
32             System.exit(0);
33          }
34       });
35       setLayout(null);
36       this.setVisible(true);
37    }
38    public static void main(String[] args) {              //add
39       new Ex1_new_ver_1();
40    }
41    public void paint(Graphics g){                        //applet の paint 部分
42       MitsuiWorld_2 m = new MitsuiWorld_2();             //インスタンス生成
43
44       m.setGraphics(bg);                                   //g → bg
45       m.setScreenSize(getSize().width,getSize().height);   //描く画面をセット
46         // [a, b] × [c, d]
47       bg.setColor(Color.white);                          //add
48       bg.fillRect(0, 0, 500, 500);                       //add
49       bg.setColor(Color.black);                          //add
50       m.setArea(-1.0,1.0,-1.0,1.0);                      //描写範囲を決める
51
52       //座標軸を設定
53       m.move(-1.0,0.0); m.draw(1.0,0.0);
54       m.move(0.0,1.0); m.draw(0.0,-1.0);
55
56       m.setColor(Color.red);                           //色を赤にする
```

```
57
58          double h=0.01;                                    //sin(π x) を 0.01 刻みで描く
59          double x=-1.0;
60          m.move(x,f(x));
61          for( x=-1.01;x<=1.0;x+=h){
62             m.draw(x,f(x));
63          }
64          g.drawImage(im, 0, 0, this);                        //add
65          saveJpeg();                                         //add
66       }
67
68       public double f(double x){
69          return Math.sin(Math.PI*x);
70       }
71    }
```

A.4 wave2d_simple_1.java

2 次元波動方程式を時間変化と共に描いていくプログラムです。
波動方程式

$$\frac{1}{c^2}u_{tt} = u_{xx} + u_{yy} \ \ (t > 0, (x, y) \in \Omega) \ , \ \Omega = \{(x, y); -2 < x < 2, -2 < y < 2\}$$

と初期条件

$$u(x, y, 0) = \phi(x, y), u_t(x, y, 0) = \psi(x, y) \ \ ((x, y) \in \bar{\Omega})$$

と境界条件
(Neumann 境界条件) $\left.\dfrac{\partial u}{\partial n}(x, y, t)\right|_{\partial\Omega} = 0 \ (t > 0)$
からなる初期値境界値問題です。
初期値を

$$\phi(x, y) = \frac{\sin \pi x}{2} + \frac{\sin \pi y}{2}$$
$$\psi(x, y) = 0$$

としています。
次の様にしてコンパイルします。

```
javac wave2d_simple_1.java
```

実行は

```
appletviewer wave2d_simple_1.java &
```

```
1    /*
2     *<applet code ="wave2d_simple_1.class" width=500 height=500></applet>
3     *
4     *三井さんの Wave2d の
5     *func と bc を 1 パターンのみに。
6     */
```

```
 7
 8  import java.applet.*;
 9  import java.awt.*;
10  import Mitsui.*;
11
12  public class wave2d_simple_1 extends Applet {
13      MitsuiWorld_2 m = new MitsuiWorld_2();
14      double xmin, xmax, ymin, ymax, zmin, zmax;
15      public void init(){
16          Graphics g= getGraphics();
17          m.setGraphics(g);
18          m.setScreenSize(getSize().width, getSize().height);
19          xmin = -2.0;
20          xmax = 2.0;
21          ymin = -2.0;
22          ymax = 2.0;
23          zmin = -1.0;
24          zmax = 1.0;
25          m.setArea2(xmin, xmax, ymin, ymax, zmin, zmax);
26          m.setColor(3);
27      }
28      public void paint(Graphics g) {
29          int nx = 40;
30          int ny = 40;
31          double tau = 0.05;
32          double Tmax = 4.0;
33          double dx = (xmax-xmin)/nx;
34          double dy = (ymax-ymin)/ny;
35          double lambdax = tau / dx;
36          double lambday = tau / dy;
37          double lambdax2 = lambdax * lambdax;
38          double lambday2 = lambday * lambday;
39          double[][] u1 = new double[nx+1][ny+1];
40          double[][] u2 = new double[nx+1][ny+1];
41          double[][] u3 = new double[nx+1][ny+1];
42          for(int i=0; i<=nx; i++)
43                  for(int j=0; j<=ny; j++)
44                      u1[i][j] = phi(xmin+i*dx,ymin+j*dy);
45          for(int i=1; i<nx; i++)
46                  for(int j=1; j<ny; j++)
47                      u2[i][j] = (1.0-lambdax2-lambday2) * u1[i][j] +
48                          0.5 * lambdax2 * (u1[i-1][j]+u1[i+1][j]) +
49                          0.5 * lambday2 * (u1[i][j-1]+u1[i][j+1]) +
50                          tau * psi(xmin+i*dx,ymin+j*dy);
51                  //Neumann 境界条件
52                  for(int i=0;i<=nx;i++){
53                      u2[i][0]=u2[i][1];
54                      u2[i][ny]=u2[i][ny-1];
55                  }
56                  for(int i=0;i<=ny;i++){
57                      u2[0][i]=u2[1][i];
```

```
58                      u2[nx][i]=u2[nx-1][i];
59                  }
60              m.hideBirdView(u1, nx, ny, 1);
61                g.clearRect(0, 30, getSize().width, getSize().height);
62                m.hideBirdView(u2, nx, ny, 1);
63
64                int kmax = (int)Math.rint(Tmax/ tau);
65               for(int k= 1; k<= kmax; k++){
66                    for(int i=1;i<nx;i++)
67                     for(int j=1;j<ny;j++)
68                      u3[i][j] = 2 * (1.0-lambdax2 -lambday2) * u2[i][j] +
69                          lambdax2 * (u2[i-1][j]+u2[i+1][j]) +
70                          lambday2 * (u2[i][j-1]+u2[i][j+1]) - u1[i][j];
71                  //Neumann 境界条件
72                  for(int i=0;i<=nx;i++){
73                      u3[i][0] = u3[i][1] ;
74                      u3[i][ny]= u3[i][ny-1];
75                  }
76                  for(int i=0;i<=ny;i++){
77                      u3[0][i] = u3[1][i];
78                      u3[nx][i]= u3[nx-1][i];
79                  }
80              g.clearRect(0, 30, getSize().width, getSize().height);
81                m.hideBirdView(u3, nx, ny, 1);
82              for(int i=0;i<=nx;i++){
83                  for(int j=0;j<=ny;j++){
84                      u1[i][j] = u2[i][j];
85                      u2[i][j] = u3[i][j];
86                  }
87              }
88           }
89      }
90      public double phi(double x,double y){
91           return Math.sin(Math.PI*x)/2+Math.sin(Math.PI*y)/2;
92      }
93      public double psi(double x,double y){
94           return 0.0;
95      }
96   }
```

A.5 wave2d_simple_new_ver_1.java

wave2d_simple_1.java のプログラムを修正し、描いたグラフを逐次、画像ファイルの Jpeg 形式に記録出来る様にしたプログラムです。
次の様にしてコンパイルします。

```
javac wave2d_simple_new_ver_1.java
```

実行は

```
java wave2d_simple_new_ver_1
```

```
1   /*
2    *三井さんの Wave2d の
3    *func と bc を 1 パターンのみに。
4    *jpeg ファイル出力できるように。
5    *コンパイルは javac wave2d_simple_new_ver_1.java
6    *実行は java wave2d_simple_new_ver_1
7    */
8
9   // import java.applet.*;
10  import java.awt.*;
11  import Mitsui.*;
12  import java.awt.event.*;              //add
13  import java.awt.image.BufferedImage;  //add
14  import java.io.File;                  //add
15  import javax.imageio.ImageIO;         //add
16
17  public class wave2d_simple_new_ver_1 extends Frame {          //Applet → Frame
18    MitsuiWorld_2 m = new MitsuiWorld_2();
19    double xmin, xmax, ymin, ymax, zmin, zmax;
20    BufferedImage im = new BufferedImage(500, 500, BufferedImage.TYPE_3BYTE_BGR);//add
21    Graphics bg = im.getGraphics();                    //add
22    private boolean result = false;                    //add
23    private int jpeg_file_number = 0;                  //add
24    void saveJpeg() {                                  //add
25      String str;
26      if (jpeg_file_number < 10)
27        str = "0000" + jpeg_file_number;
28      else if (jpeg_file_number < 100)
29        str = "000" + jpeg_file_number;
30      else if (jpeg_file_number < 1000)
31        str = "00" + jpeg_file_number;
32      else if (jpeg_file_number < 10000)
33        str = "0" + jpeg_file_number;
34      else
35        str = "" + jpeg_file_number;
36      try {
37        result = ImageIO.write(im, "jpeg", new File("test" + str + ".jpg"));
38        jpeg_file_number++;
39      }
40      catch (Exception e) {
41        e.printStackTrace();
42        result = false;
43      }
44    }
45    public wave2d_simple_new_ver_1() {                          //add
46      this.setSize(500, 500);
47      this.addWindowListener(new WindowAdapter() {
48        public void windowClosing(WindowEvent ev) {
```

```
49          System.exit(0);
50        }
51      });
52      setLayout(null);
53      this.setVisible(true);
54      init();
55    }
56    public static void main(String[] args) {               //add
57      new wave2d_simple_new_ver_1();
58    }
59    public void init(){
60  //     Graphics g= getGraphics();                        //erase
61       m.setGraphics(bg);                                  //g → bg
62       m.setScreenSize(getSize().width, getSize().height);
63       xmin = -2.0;
64       xmax = 2.0;
65       ymin = -2.0;
66       ymax = 2.0;
67       zmin = -1.0;
68       zmax = 1.0;
69       m.setArea2(xmin, xmax, ymin, ymax, zmin, zmax);
70       m.setColor(3);
71    }
72    public void paint(Graphics g) {
73       int nx = 40;
74       int ny = 40;
75       double tau = 0.05;
76       double Tmax = 4.0;
77       double dx = (xmax-xmin)/nx;
78       double dy = (ymax-ymin)/ny;
79       double lambdax = tau / dx;
80       double lambday = tau / dy;
81       double lambdax2 = lambdax * lambdax;
82       double lambday2 = lambday * lambday;
83       double[][] u1 = new double[nx+1][ny+1];
84       double[][] u2 = new double[nx+1][ny+1];
85       double[][] u3 = new double[nx+1][ny+1];
86       for(int i=0; i<=nx; i++)
87          for(int j=0; j<=ny; j++)
88       u1[i][j] = phi(xmin+i*dx,ymin+j*dy);
89       for(int i=1; i<nx; i++)
90          for(int j=1; j<ny; j++)
91            u2[i][j] = (1.0-lambdax2-lambday2) * u1[i][j] +
92            0.5 * lambdax2 * (u1[i-1][j]+u1[i+1][j]) +
93            0.5 * lambday2 * (u1[i][j-1]+u1[i][j+1]) +
94            tau * psi(xmin+i*dx,ymin+j*dy);
95       //Neumann 境界条件
96       for(int i=0;i<=nx;i++){
97          u2[i][0]=u2[i][1];
98          u2[i][ny]=u2[i][ny-1];
99       }
```

```
100         for(int i=0;i<=ny;i++){
101            u2[0][i]=u2[1][i];
102            u2[nx][i]=u2[nx-1][i];
103         }
104         m.hideBirdView(u1, nx, ny, 1);
105   //     g.clearRect(0, 30, getSize().width, getSize().height);  //erase
106         bg.setColor(Color.white);                              //add
107         bg.fillRect(0, 0, 500, 500);                           //add
108         bg.setColor(Color.black);                              //add
109         m.hideBirdView(u2, nx, ny, 1);
110         g.drawImage(im, 0, 0, this);                           //add
111         saveJpeg();                                            //add
112
113         int kmax = (int)Math.rint(Tmax/ tau);
114         for(int k= 1; k<= kmax; k++){
115            for(int i=1;i<nx;i++)
116               for(int j=1;j<ny;j++)
117                  u3[i][j] = 2 * (1.0-lambdax2 -lambday2) * u2[i][j] +
118                             lambdax2 * (u2[i-1][j]+u2[i+1][j]) +
119                             lambday2 * (u2[i][j-1]+u2[i][j+1]) - u1[i][j];
120            //Neumann 境界条件
121            for(int i=0;i<=nx;i++){
122               u3[i][0] = u3[i][1] ;
123               u3[i][ny]= u3[i][ny-1];
124            }
125            for(int i=0;i<=ny;i++){
126               u3[0][i] = u3[1][i];
127               u3[nx][i]= u3[nx-1][i];
128            }
129   //         g.clearRect(0, 30, getSize().width, getSize().height);  //erase
130            bg.setColor(Color.white);                              //add
131            bg.fillRect(0, 0, 500, 500);                           //add
132            bg.setColor(Color.black);                              //add
133            m.hideBirdView(u3, nx, ny, 1);
134            g.drawImage(im, 0, 0, this);                           //add
135            saveJpeg();                                            //add
136            for(int i=0;i<=nx;i++){
137               for(int j=0;j<=ny;j++){
138                  u1[i][j] = u2[i][j];
139                  u2[i][j] = u3[i][j];
140               }
141            }
142         }
143      }
144      public double phi(double x,double y){
145         return Math.sin(Math.PI*x)/2+Math.sin(Math.PI*y)/2;
146      }
147      public double psi(double x,double y){
148         return 0.0;
149      }
150   }
```

参考文献

[1] 三井 康之, Java による波動方程式の数値解析, 2001 年度桂田研卒業研究, http://www.math.meiji.ac.jp/~mk/labo/pdf/2001-mitsui.pdf

[2] 伊藤 秀範, Java による波動方程式のシミュレーションプログラム ～ 1 次元, 2 次元の波の動き, 2004 年度桂田研卒業研究, http://www.math.meiji.ac.jp/~mk/labo/report/open/2004-itou.pdf