2004年度　卒業研究

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～

明治大学理工学部数学科
4年16組7番　伊藤　秀範

2005 年2 月25 日

はじめに

私は２００１年度卒業された三井康之さんの卒業研究の Java のプログラムを改良し、１次元、２次元の平面波、定常波をシミュレートできるようにすることと、２次元の場合に等高線表示ができるようにすることの２つを目標にして研究しました。

このレポートは `http://www.math.meiji.ac.jp/~mk/labo/report/` で公開しています。是非興味のある方は見てください。

Java はプラットホームに依存しないので１度プログラムを作ればグラフィックスを用いるプログラムでも、いろいろなソフトウェア環境で実行することが可能になります。今回作製したアプレットを用いることにより、たくさんの人に容易に数値解析を体感してもらいたいです。

最後に私の卒業研究を熱心にご指導をしてくださった桂田祐史先生に心より深く感謝します。

目 次

第 1 章　１次元波動方程式 **3**
1.1　差分解 3
1.2　境界条件 5
1.2.1　Dirichlet 境界条件 5
1.2.2　Neumann 境界条件の素朴な差分近似 5
1.2.3　Neumann 境界条件の仮想格子点を用いる差分近似 6

第 2 章　２次元波動方程式 **7**
2.1　差分解 7
2.2　境界条件 10
2.2.1　Dirichlet 境界条件 10
2.2.2　Neumann 境界条件の素朴な差分近似 10
2.2.3　Neumann 境界条件の仮想格子点を用いた差分近似 10

第 3 章　波の動きについて〜平面波、定常波〜 **14**
3.1　１次元の波について〜平面波、定常波〜 14
3.2　２次元の波について〜平面波、定常波〜 16

第 4 章　Java によるシミュレーションプログラミング **18**
4.1　１次元、２次元方程式の program〜`WaveAll_2_1.java`〜 18
4.2　等高線についての program〜`toukousen3.java`〜 42

第1章　1次元波動方程式

1.1　差分解

波動方程式

$$\frac{1}{c^2}u_{tt} = u_{xx} \qquad (t > 0,\ x \in (0,1)) \tag{1.1}$$

と初期条件

$$u(x,0) = \phi(x) \tag{1.2}$$
$$u_t(x,0) = \psi(x) \qquad (x \in [0,1]) \tag{1.3}$$

それと次のいづれかの境界条件

$$\text{(Dirichlet 境界条件)} \qquad u(0,t) = u(1,t) = 0 \qquad (t > 0) \tag{1.4}$$
$$\text{(Neumann 境界条件)} \qquad u_x(0,t) = u_x(1,t) = 0 \qquad (t > 0) \tag{1.5}$$

からなる初期値境界値問題に対する差分方程式を求める。

まず、区間 $[0,1]$ を N 等分し、格子点を x_i と表す。すなわち

$$h = \frac{1}{N}$$

として

$$x_i = ih \qquad (i = 0,1,2,\cdots).$$

一方、$\tau > 0$ を固定して、

$$t_j = j\tau \qquad (j = 0,1,2,\cdots)$$

とおく。格子点 (x_i, t_j) での u の値を u_{ij} とおく:$u_{ij} = u(x_i, t_j)$、u_{ij} において偏導関数 u_{tt},u_{xx} をそれぞれ2階中心差分近似すると、

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

であることから、式 (1.1) より

$$\frac{1}{c^2}\frac{U_i^{j+1}-2U_i^j+U_i^{j-1}}{\tau^2}=\frac{U_{i+1}^j-2U_i^j+U_{i-1}^j}{h^2}$$

なる差分方程式を得る。ただし U_i^j は $u(x_i,t_j)$ の近似値である。

両辺に τ^2 をかけてから移項すると

$$\begin{aligned} U_i^{j+1} &= 2(1-\lambda^2)U_i^j+\lambda^2(U_{i+1}^j+U_{i-1}^j)-U_i^{j-1} \qquad \cdots\dagger \\ &\qquad (i=1,2,\cdots,N-1;\ j=1,2,\cdots). \end{aligned}$$

ただし、$\lambda=\dfrac{c\tau}{h}$ とした。

初期条件からは

$$U_i^0=\phi(x_i) \qquad \cdots\dagger$$

を得る。

また、$u(x,t)$ の滑らかさを C^3 級として、$t=0$ を中心としてテイラー展開すると、

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

$t=0$ でも波動方程式が成り立っていると仮定すると、

$$\begin{aligned} u(x_i,t_1) &= u(x_i,0)+\psi(x_i)\tau+c^2\frac{u_{xx}(x_i,0)}{2!}\tau^2+O(\tau^3) \\ &= u(x_i,0)+\psi(x_i)\tau+\frac{c^2}{2!}\frac{u_{i+1,0}-2u_{i,0}+u_{i-1,0}}{h^2}\tau^2+O(h^2)+O(\tau^3) \end{aligned}$$

ゆえに

$$U_i^1=U_i^0+\psi(ih)\tau+\frac{c^2}{2!}\frac{U_{i+1}^0-2U_i^0+U_{i-1}^0}{h^2}\tau^2$$

なる差分方程式を得る。

整理して

$$U_i^1=(1-\lambda^2)U_i^0+\psi(ih)\tau+\frac{\lambda^2}{2}(U_{i+1}^0+U_{i-1}^0). \qquad \cdots\dagger$$

以上から、式 (1.1),(1.2),(1.3) に対応して、未知数列 $\{U_i^j;0\le i\le N,j\ge 0\}$ に関する方程式系

$$U_i^{j+1} = 2(1-\lambda^2)U_i^j+\lambda^2(U_{i+1}^j+U_{i-1}^j)-U_i^{j-1} \tag{1.6}$$

$$
\begin{aligned}
& & & (1 \le i \le N-1, j \ge 1), \\
U_i^0 &= \phi(x_i) & & (0 \le i \le N), \qquad (1.7) \\
U_i^1 &= (1-\lambda^2)U_i^0 + \psi(ih)\tau + \frac{\lambda^2}{2}(U_{i+1}^0 + U_{i-1}^0) & & \qquad (1.8) \\
& & & (1 \le i \le N-1)
\end{aligned}
$$

が得られた。これらの式と境界条件により私たちは波動方程式の差分解を得ることができる。境界条件については次の節で述べる。

1.2 境界条件

1.2.1 Dirichlet 境界条件

境界条件

$$u(0,t) = u(1,t) = 0 \qquad (t > 0)$$

から

$$U_0^j = U_1^j = 0 \qquad (j = 1, 2, \cdots). \tag{1.9}$$

1.2.2 Neumann 境界条件の素朴な差分近似

境界条件は

$$u_x(0,t) = u_x(1,t) = 0 \qquad (t > 0)$$

で与えられ $u_x(0,t)$ については前進差分近似を、$u_x(1,t)$ については後退差分近似することを考えると、

$$
\begin{aligned}
u_x(0,t_j) &= u_x(x_0,t_j) = \frac{u(x_1,t_j) - u(x_0,t_j)}{h} + O(h), \\
u_x(1,t_j) &= u_x(x_N,t_j) = \frac{u(x_N,t_j) - u(x_{N-1},t_j)}{h} + O(h)
\end{aligned}
$$

となり

$$\frac{U_1^j - U_0^j}{h} = \frac{U_N^j - U_{N-1}^j}{h} = 0$$

なる差分方程式を得る。すなわち

$$U_0^j = U_1^j, \qquad U_N^j = U_{N-1}^j \qquad (j = 1, 2, \cdots). \tag{1.10}$$

1.2.3 Neumann 境界条件の仮想格子点を用いる差分近似

境界条件は Neuamm 境界条件とする。しかし仮想格子点 $x_{-1},\ x_{N+1}$ を考え、$u_x(0,t),\ u_x(1,t)$ をともに 1 階の中心差分近似することで、境界条件に対応する差分方程式として

$$U^j_{-1} = U^j_1, \quad U^j_{N-1} = U^j_{N+1}$$

を得る。

さらに式 (1.6) と (1.8) が $i = 0,\ i = N$ でも成り立つとすると、式 (1.8) は

$$\begin{aligned} U^1_0 &= (1-\lambda^2)U^0_0 + \psi(ih)\tau + \lambda^2 U^0_1, && (1.11)\\ U^1_N &= (1-\lambda^2)U^0_N + \psi(ih)\tau + \lambda^2 U^0_{N-1} && (1.12) \end{aligned}$$

となり、式 (1.6) は

$$\begin{aligned} U^{j+1}_0 &= 2(1-\lambda^2)U^j_0 + 2\lambda^2 U^j_1 - U^j_0, && (1.13)\\ U^{j+1}_N &= 2(1-\lambda^2)U^j_N + 2\lambda^2 U^j_{N-1} - U^j_N && (1.14) \end{aligned}$$

となる。

第2章　2次元波動方程式

2.1　差分解

波動方程式

$$\frac{1}{c^2}u_{tt} = u_{xx} + u_{yy} \quad (t > 0,(x,y) \in \Omega) \tag{2.1}$$

$$\Omega = \{(x,y); A < x < B, C < y < D\}$$

と初期条件

$$u(x,y,0) = \phi(x,y), \tag{2.2}$$

$$u_t(x,y,0) = \psi(x,y) \quad ((x,y) \in \overline{\Omega}) \tag{2.3}$$

と次の境界条件のいずれか

$$\text{(Dirichlet 境界条件)} \qquad u(x,y,t)|_{\partial\Omega} = 0 \quad (t > 0), \tag{2.4}$$

$$\text{(Neumann 境界条件)} \qquad \frac{\partial u}{\partial n}(x,y,t)|_{\partial\Omega} = 0 \quad (t > 0) \tag{2.5}$$

からなる初期値境界値問題に対する差分方程式を求める。

まず領域Ωを座標軸に平行な格子線で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

$$x_i = A + id_x \quad (i = 0,1,2,\cdots,N_x),$$

$$y_j = C + jd_y \quad (j = 0,1,2,\cdots,N_y)$$

と置く。次に$\tau > 0$を固定して

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

と置く。格子点 (x_i, y_j, t_k) において偏導関数 u_{tt},u_{xx},u_{yy} をそれぞれ 2 階中心差分近似すると

$$\begin{aligned} u_{tt}(x_i, y_j, t_k) &= \frac{u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}}{\tau^2} + O(\tau^2) \\ u_{xx}(x_i, y_j, t_k) &= \frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{d_x^2} + O(d_x^2) \\ u_{yy}(x_i, y_j, t_k) &= \frac{u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}}{d_y^2} + O(d_y^2) \end{aligned}$$

となる。ただし $u_{ijk} = u(x_i, y_j, t_k)$ とする。ここで U_{ij}^k は u_{ijk} を近似されると期待される量である。式 (2.1) は

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

を得る。

両辺に τ^2 をかけてから移項すると、

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

$$(i = 1, 2, \cdots, N_x - 1; j = 1, 2, \cdots, N_y - 1; k = 1, 2, \cdots)$$

が導かれる。

また初期条件より

$$U_{i,j}^0 = \phi(x_i, y_j). \qquad \cdots\dagger$$

$u(x, y, t)$ の滑らかさを C^3 級と仮定し、$u(x_i, y_j, t)$ を $t = 0$ を中心としてテイラー展開すると、

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

$t = 0$ でも式 (2.1) が成り立つとすると

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

整頓すると

$$U^1_{i,j} = U^0_{i,j}+\psi(A+id_x, C+jd_y)\tau+\frac{c^2}{2!}\frac{U^0_{i+1,j}-2U^0_{i,j}+U^0_{i-1,j}}{d_x^2}\tau^2+\frac{c^2}{2!}\frac{U^0_{i,j+1}-2U^0_{i,j}+U^0_{i,j-1}}{d_y^2}\tau^2.$$

よって

$$U^1_{i,j} = (1-\lambda_x^2-\lambda_y^2)U^0_{i,j}+\psi(A+id_x, C+jd_y)\tau+\frac{\lambda_x^2}{2}(U^0_{i+1,j}+U^0_{i-1,j})+\frac{\lambda_y^2}{2}(U^0_{i,j+1}+U^0_{i,j-1}). \quad \cdots\dagger$$

以上式 (2.1), (2.2), (2.3) に対応して、未知数列 $\{U^k_{i,j}; 0 \le i \le N_x, 0 \le j \le N_y, k = 1, 2, \cdots\}$ に関する方程式系

$$\begin{aligned}
U^{k+1}_{i,j} &= 2(1-\lambda_x^2-\lambda_y^2)U^k_{i,j}+\lambda_x^2(U^k_{i+1,j}+U^k_{i-1,j}), +\lambda_y^2(U^k_{i,j+1}+U^k_{i,j-1})-U^{k-1}_{i,j} \\
&\qquad (i = 1, 2, \cdots, N_x-1; j = 1, 2, \cdots, N_y-1; k = 1, 2, \cdots), \qquad (2.6)\\
U^0_{i,j} &= \phi(x_i, y_j) \qquad (0 \le i \le N_x,\ 0 \le j \le N_y), \qquad (2.7)\\
U^1_{i,j} &= (1-\lambda_x^2-\lambda_y^2)U^0_{i,j}+\psi(x_i, y_j)\tau+\frac{\lambda_x^2}{2}(U^0_{i+1,j}+U^0_{i-1,j})+\frac{\lambda_y^2}{2}(U^0_{i,j+1}+U^0_{i,j-1}) \\
&\qquad (0 \le i \le N_x-1, 0 \le j \le N_y-1) \qquad (2.8)
\end{aligned}$$

が得られた。これらの式と境界条件により私たちは波動方程式の差分解を得ることができる。境界条件は次の節で述べる。

2.2 境界条件

2.2.1 Dirichlet 境界条件

境界条件より

$$u(x,y,t)|_{\partial\Omega} = 0 \qquad (t>0)$$

$$U^k_{i,j} = 0 \quad (i=0 \text{ or } i=N_x \text{ or } j=0 \text{ or } j=N_y; k=1,2,\cdots).$$

言い換えると

$$U^k_{0,j} = U^k_{Nx,j} = 0 \qquad (j=0,1,2,\cdots,N_y, k=1,2,\cdots), \tag{2.9}$$

$$U^k_{i,0} = U^k_{i,Ny} = 0 \qquad (i=0,1,2,\cdots,N_x, k=1,2,\cdots). \tag{2.10}$$

2.2.2 Neumann 境界条件の素朴な差分近似

境界条件は

$$\frac{\partial u}{\partial n}(x,y,t) = 0 \quad (t>0, (x,y)\in\partial\Omega)$$

で与えられ $u_x(x_0,y_j,t_k)$, $u_y(x_i,y_0,t_k)$ については前進差分近似を、$u_x(x_{N_x},y_j,t_k)$, $u_y(x_i,y_{N_y},t_k)$ については後退差分近似することを考えると、

$$\frac{U^k_{1,j}-U^k_{0,j}}{d_x} = 0, \quad \frac{U^k_{N_x,j}-U^k_{N_x-1,j}}{d_x} = 0,$$

$$\frac{U^k_{i,1}-U^k_{i,0}}{d_y} = 0, \quad \frac{U^k_{i,N_y}-U^k_{i,N_y-1}}{d_y} = 0.$$

言い換えると

$$U^k_{1,j} = U^k_{0,j}, \quad U^k_{N_x,j} = U^k_{N_x-1,j}, \tag{2.11}$$

$$U^k_{i,1} = U^k_{i,0}, \quad U^k_{i,N_y} = U^k_{i,N_y-1}. \tag{2.12}$$

2.2.3 Neumann 境界条件の仮想格子点を用いた差分近似

ここでも境界条件はNeunamm境界条件とする。しかし仮想格子点(x_{-1},y_j),(x_{Nx+1},y_j),(x_i,y_{-1}), (x_i,y_{Ny+1}) が存在するとして $u_x(A,y,t)$,$u_x(B,y,t)$,$u_y(x,C,t)$,$u_y(x,D,t)$ をともに
１階の中心差分近似することを考えると、境界条件に対応する差分方程式として

$$U^k_{-1,j} = U^k_{1,j}, \quad U^k_{N_x+1,j} = U^k_{N_x-1,j},$$

$$U^k_{i,-1} = U^k_{i,1}, \quad U^k_{i,N_y+1} = U^k_{i,N_y-1}$$

を得る。

さらに、式 (2.6),(2.8) がそれぞれ $i=0,\quad i=N_x,\quad j=0,\quad j=N_y$ でも成り立つとする。

(2.8) 式に、まず $j=0$ を代入すると

$$U^1_{i,0}=\tau\psi(x_i,y_0)+(1-\lambda_x^2-\lambda_y^2)U^0_{i,0}+\frac{1}{2}\lambda_x^2(U^0_{i+1,0}+U^0_{i-1,0})+\frac{1}{2}\lambda_y^2(2U^0_{i,1})$$

となる。この式で $i=0$ とすると $U^0_{-1,0}$ が、$i=N_x$ とすると $U^0_{N_x+1,0}$ がでてくる。そこで i の値によって場合わけすると、

$i=0$ のとき、

$$U^1_{0,0} = \tau\psi(x_0,y_0)+(1-\lambda_x^2-\lambda_y^2)U^0_{0,0}+\frac{1}{2}\lambda_x^2(2U^0_{1,0})+\frac{1}{2}\lambda_y^2(2U^0_{0,1}),$$

$i=1,2,\cdots,N_x-1$ のとき、

$$U^1_{i,0} = \tau\psi(x_i,y_0)+(1-\lambda_x^2-\lambda_y^2)U^0_{i,0}+\frac{1}{2}\lambda_x^2(U^0_{i+1,0}+U^0_{i-1,0})+\frac{1}{2}\lambda_y^2(2U^0_{i,1}),$$

$i=N_x$のとき、

$$U^1_{N_x,0} = \tau\psi(x_{N_x},y_0)+(1-\lambda_x^2-\lambda_y^2)U^0_{N_x,0}+\frac{1}{2}\lambda_x^2(2U^0_{N_x-1,0})+\frac{1}{2}\lambda_y^2(2U^0_{N_x,1})$$

となる。

また、(2.8) 式に $j=N_y$ を代入すると

$$U^1_{i,N_y}=\tau\psi(x_i,y_{N_y})+(1-\lambda_x^2-\lambda_y^2)U^0_{i,N_y}+\frac{1}{2}\lambda_x^2(U^0_{i+1,N_y}+U^0_{i-1,N_y})+\frac{1}{2}\lambda_y^2(2U^0_{i,N_y-1})$$

となり、ここでも $i=0$ で $U^0_{-1,0}$ が、$i=N_x$ で $U^0_{N_x+1,0}$ がでてくるので、i の値によって場合わけすると

$i=0$ のとき、

$$U^1_{0,N_y} = \tau\psi(x_0,y_{N_y})+(1-\lambda_x^2-\lambda_y^2)U^0_{0,N_y}+\frac{1}{2}\lambda_x^2(2U^0_{1,N_y})+\frac{1}{2}\lambda_y^2(2U^0_{0,N_y-1}),$$

$i=1,2,\cdots,N_x-1$ のとき、

$$U^1_{i,N_y} = \tau\psi(x_i,y_{N_y})+(1-\lambda_x^2-\lambda_y^2)U^0_{i,N_y}+\frac{1}{2}\lambda_x^2(U^0_{i+1,N_y}+U^0_{i-1,N_y})+\frac{1}{2}\lambda_y^2(2U^0_{i,N_y-1}),$$

$i=N_x$のとき、

$$U^1_{N_x,N_y} = \tau\psi(x_{N_x},y_{N_y})+(1-\lambda_x^2-\lambda_y^2)U^0_{N_x,N_y}+\frac{1}{2}\lambda_x^2(2U^0_{N_x-1,N_y})+\frac{1}{2}\lambda_y^2(2U^0_{N_x,N_y-1}).$$

次は $i=0$ を式 (2.8) に代入すると、

$$U^1_{0,j}=\tau\psi(x_0,y_j)+(1-\lambda_x^2-\lambda_y^2)U^0_{0,j}+\frac{1}{2}\lambda_x^2(2U^0_{1,j})+\frac{1}{2}\lambda_y^2(U^0_{0,j+1}+U^0_{0,j-1})$$

となり、ここでは $j=0$ で $U^0_{0,-1}$ が、$j=N_y$ で U^0_{0,N_y+1} がでてくるので、j の値によって場合わけすると

$j=0$ のとき、

$$U^1_{0,0} = \tau\psi(x_0,y_0)+(1-\lambda_x^2-\lambda_y^2)U^0_{0,0}+\frac{1}{2}\lambda_x^2(2U^0_{1,0})+\frac{1}{2}\lambda_y^2(2U^0_{0,1}),$$

$j=1,2,\cdots,N_x-1$ のとき、

$$U^1_{0,j} = \tau\psi(x_0,y_j)+(1-\lambda_x^2-\lambda_y^2)U^0_{0,j}+\frac{1}{2}\lambda_x^2(2U^0_{1,j})+\frac{1}{2}\lambda_y^2(U^0_{0,j+1}+U^0_{0,j-1}),$$

$j=N_y$のとき、

$$U^1_{0,N_y} = \tau\psi(x_0,y_{N_y})+(1-\lambda_x^2-\lambda_y^2)U^0_{0,N_y}+\frac{1}{2}\lambda_x^2(2U^0_{1,N_y})+\frac{1}{2}\lambda_y^2(2U^0_{0,N_y-1}).$$

また $i=N_x$ を式 (2.8) に代入すると、

$$U^1_{N_x,j} = \tau\psi(x_{N_x},y_j)+(1-\lambda_x^2-\lambda_y^2)U^0_{N_x,j}+\frac{1}{2}\lambda_x^2(2U^0_{N_x-1,j})+\frac{1}{2}\lambda_y^2(U^0_{N_x,j+1}+U^0_{N_x,j-1})$$

となり、これも j で場合わけすると

$j=0$ のとき、

$$U^1_{N_x,0} = \tau\psi(x_{N_x},y_0)+(1-\lambda_x^2-\lambda_y^2)U^0_{N_x,0}+\frac{1}{2}\lambda_x^2(2U^0_{N_x-1,0})+\frac{1}{2}\lambda_y^2(2U^0_{N_x,1}),$$

$j=1,2,\cdots,N_x-1$ のとき、

$$U^1_{N_x,j} = \tau\psi(x_{N_x},y_j)+(1-\lambda_x^2-\lambda_y^2)U^0_{N_x,j}+\frac{1}{2}\lambda_x^2(2U^0_{N_x-1,j})+\frac{1}{2}\lambda_y^2(U^0_{N_x,j+1}+U^0_{N_x,j-1}),$$

$j=N_y$のとき、

$$U^1_{N_x,N_y} = \tau\psi(x_{N_x},y_{N_y})+(1-\lambda_x^2-\lambda_y^2)U^0_{N_x,N_y}+\frac{1}{2}\lambda_x^2(2U^0_{N_x-1,N_y})+\frac{1}{2}\lambda_y^2(2U^0_{N_x,N_y-1}).$$

以上の式を見てみると $U^1_{0,0}$,$U^1_{N_x,0}$,U^1_{0,N_y},$U^1_{N_x,N_y}$ の４つの式がダブっていいる。これらを一つにまとめて整理すると式 (2.8) は

$$U^1_{i,0} = \tau\psi(x_i,y_0)+(1-\lambda_x^2-\lambda_y^2)U^0_{i,0}+\frac{1}{2}\lambda_x^2(U^0_{i+1,0}+U^0_{i-1,0})+\frac{1}{2}\lambda_y^2(2U^0_{i,1}) \quad (i=1,2,\cdots,N_x-1), \quad (2.13)$$

$$U^1_{i,N_y} = \tau\psi(x_i,y_{N_y})+(1-\lambda_x^2-\lambda_y^2)U^0_{i,N_y}+\frac{1}{2}\lambda_x^2(U^0_{i+1,N_y}+U^0_{i-1,N_y})+\frac{1}{2}\lambda_y^2(2U^0_{i,N_y-1}) \quad (i=1,2,\cdots,N_x-1), \quad (2.14)$$

$$U^1_{0,j} = \tau\psi(x_0,y_j)+(1-\lambda_x^2-\lambda_y^2)U^0_{0,j}+\frac{1}{2}\lambda_x^2(2U^0_{1,j})+\frac{1}{2}\lambda_y^2(U^0_{0,j+1}+U^0_{0,j-1}) \quad (j=1,2,\cdots,N_x-1), \quad (2.15)$$

$$U^1_{N_x,j} = \tau\psi(x_{N_x},y_j)+(1-\lambda_x^2-\lambda_y^2)U^0_{N_x,j}+\frac{1}{2}\lambda_x^2(2U^0_{N_x-1,j})+\frac{1}{2}\lambda_y^2(U^0_{N_x,j+1}+U^0_{N_x,j-1})$$
$$(j=1,2,\cdots,N_x-1), \quad (2.16)$$

$$U^1_{0,0} = \tau\psi(x_0,y_0)+(1-\lambda_x^2-\lambda_y^2)U^0_{0,0}+\frac{1}{2}\lambda_x^2(2U^0_{1,0}+\frac{1}{2}\lambda_y^2(2U^0_{0,1})$$
$$(i=0,j=0), \quad (2.17)$$

$$U^1_{N_x,0} = \tau\psi(x_{N_x},y_0)+(1-\lambda_x^2-\lambda_y^2)U^0_{N_x,0}+\frac{1}{2}\lambda_x^2(2U^0_{N_x-1,0})+\frac{1}{2}\lambda_y^2(2U^0_{N_x,1})$$
$$(i=N_x,j=0), \quad (2.18)$$

$$U^1_{0,N_y} = \tau\psi(x_0,y_{N_y})+(1-\lambda_x^2-\lambda_y^2)U^0_{0,N_y}+\frac{1}{2}\lambda_x^2(2U^0_{1,N_y})+\frac{1}{2}\lambda_y^2(2U^0_{0,N_y-1})$$
$$(i=0,j=N_y), \quad (2.19)$$

$$U^1_{N_x,N_y} = \tau\psi(x_{N_x},y_{N_y})+(1-\lambda_x^2-\lambda_y^2)U^0_{N_x,N_y}+\frac{1}{2}\lambda_x^2(2U^0_{N_x-1,N_y})+\frac{1}{2}\lambda_y^2(2U^0_{N_x,N_y-1})$$
$$(i=N_x,j=N_y) \quad (2.20)$$

となる。

式 (2.6) も同様にして

$$U^{k+1}_{i,0} = 2(1-\lambda_x^2-\lambda_y^2)U^k_{i,0}+\lambda_x^2(U^k_{i+1,0}+U^k_{i-1,0})+\lambda_y^2(2U^k_{i,1})-U^{k-1}_{i,0}$$
$$(i=1,2,\cdots,N_x-1;k\geq 1), \quad (2.21)$$

$$u^{k+1}_{i,N_y} = 2(1-\lambda_x^2-\lambda_y^2)U^k_{i,N_y}+\lambda_x^2(U^k_{i+1,N_y}+U^k_{i-1,N_y})+\lambda_y^2(2U^k_{i,N_y-1})-U^{k-1}_{i,N_y}$$
$$(i=1,2,\cdots,N_x-1;k\geq 1), \quad (2.22)$$

$$U^{k+1}_{0,j} = 2(1-\lambda_x^2-\lambda_y^2)U^k_{0,j}+\lambda_x^2(2U^k_{1,j})+\lambda_y^2(U^k_{0,j+1}+U^k_{0,j-1})-U^{k-1}_{0,j}$$
$$(j=1,2,\cdots,N_y-1;k\geq 1), \quad (2.23)$$

$$U^{k+1}_{N_x,j} = 2(1-\lambda_x^2-\lambda_y^2)U^k_{N_x,j}+\lambda_x^2(2U^k_{N_x-1,j})+\lambda_y^2(U^k_{N_x,j+1}+U^k_{N_x,j-1})-U^{k-1}_{N_x,j}$$
$$(j=1,2,\cdots,N_x-1;k\geq 1), \quad (2.24)$$

$$U^{k+1}_{0,0} = 2(1-\lambda_x^2-\lambda_y^2)U^k_{0,0}+\lambda_x^2(2U^k_{1,0})+\lambda_y^2(2U^k_{0,1})-U^{k-1}_{0,0}$$
$$(i=0,j=0;k\geq 1), \quad (2.25)$$

$$U^{k+1}_{N_x,0} = 2(1-\lambda_x^2-\lambda_y^2)U^k_{N_x,0}+\lambda_x^2(2U^k_{N_x-1,0})+\lambda_y^2(2U^k_{N_x,1})-U^{k-1}_{N_x,0}$$
$$(i=N_x,j=0;k\geq 1), \quad (2.26)$$

$$U^{k+1}_{0,N_y} = 2(1-\lambda_x^2-\lambda_y^2)U^k_{0,N_y}+\lambda_x^2(2U^k_{1,N_y})+\lambda_y^2(2U^k_{0,N_y-1})-U^{k-1}_{0,N_y}$$
$$(i=0,j=N_y;k\geq 1), \quad (2.27)$$

$$U^{k+1}_{N_x,N_y} = 2(1-\lambda_x^2-\lambda_y^2)U^k_{N_x,N_y}+\lambda_x^2(2U^k_{N_x-1,N_y})+\lambda_y^2(2U^k_{N_x,N_y-1})-U^{k-1}_{N_x,N_y}$$
$$(i=N_x,j=N_y;k\geq 1) \quad (2.28)$$

となる。

第3章　波の動きについて～平面波、定常波～

3.1　１次元の波について～平面波、定常波～

波動方程式

$$\frac{1}{c^2}u_{tt} = u_{xx} \qquad (t > 0, x \in \mathbf{R})$$

を考える。$f : \mathbf{R} \to \mathbf{R}$ を任意の C^2- 級の関数とするとき、(以下 c は正の定数と仮定する)

$$u(x,t) = f(x - ct) \tag{3.1}$$

で定義される u は波動方程式を満たす。これを示すことは容易である。

$$\begin{aligned} u_t(x,t) &= f'(x-ct)(-c) \\ u_{tt}(x,t) &= f''(x-ct)(-c)^2 \\ u_x(x,t) &= f'(x-ct) \\ u_{xx}(x-ct) &= f''(x-ct) \end{aligned}$$

よって (3.1) が波動方程式を満たすことを確認できた。

初期条件

$$\begin{aligned} u(x,0) &= \phi(x), \\ u_t(x,0) &= \psi(x) \end{aligned}$$

は次のように求められる:

$$\begin{aligned} u(x,0) &= f(x - c\cdot 0) = f(x) = \phi(x), \\ u_t(x,0) &= (-c)f'(x - c\cdot 0) = (-c)f'(x) = \psi(x). \end{aligned}$$

c は、波の一定速度を示していて、$u(x,t) = f(x-ct)$ は右方向, $u(x,t) = f(x+ct)$ は左方向に進む。

後述する Java プログラムによるシミュレーションでは、f を $[0,1]$ 上定義され、台が内部 $(0,1)$ に含まれているような関数に取って実験する。0 とならないところか境界に到達するまでは、初期値問題の解を表すと期待される。

3.2　２次元の波について～平面波、定常波～

初期値境界値問題 (プログラムで扱える問題)

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

$$\Omega = \{0 < x < 4, 0 < y < 4\}$$

初期条件

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

境界条件は次のいずれかとする

$$\begin{array}{ll} \text{(Dirichlet 境界条件)} & u(x,y,t)|_{\partial\Omega} = 0 \quad (t > 0) \\ \text{(Neumann 境界条件)} & \dfrac{\partial u}{\partial n}(x,y,t)|_{\partial\Omega} = 0 \quad (t > 0) \end{array}$$

～平面波について～

$U : \mathbf{R} \to \mathbf{R}$ を滑らかな関数とし、ν を単位ベクトルとするとき、

$$u(x,y,t) = U(\nu \cdot \chi - ct), \quad \chi = (x,y)$$

とおくと、u は $\mathbf{R}$ 全体で波動方程式をみたす。これは ν の方向に速さ c で進む波 (**平面波**) を表す。

初期値は

$$\begin{aligned} u(x,y,0) &= U(\nu \cdot \chi) =: \phi(x,y), \\ u_t(x,y,0) &= -cU'(\nu \cdot \chi) =: \psi(x,y) \end{aligned}$$

である。私の数値実験では

$$c = 1, \quad U(r) = \begin{cases} \phi(x) = \frac{1}{4}(\sin \pi(2r - 0.5) + 1) & (-1 < r < 0) \\ 0 & (\text{それ以外}) \end{cases}$$

として、シミュレーションした。

- $\nu = (1,0)$ x 軸方向に進む平面波

- $\nu = (0,1)$ y 軸方向に進む平面波
- $\nu = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ 斜めに動く平面波

厳密には、平面波は初期値境界値問題の解にはならないが、境界で反射した波の影響が及ばないところでは、平面波と同じになるはずである。

~定常波について~

初期値境界値問題を Fourier の方法で解いた解の公式

$$u(x,y,t) = \sum_{m,n=1}^{\infty} \sin n\pi x \sin m\pi y (a_{nm}\cos(\sqrt{n^2+m^2}c\pi t) + b_{nm}\sin(\sqrt{n^2+m^2}c\pi t))$$
$$((x,y) \in \Omega = (0,1)\times(0,1), t > 0)$$

の一つ一つの項も波動方程式と境界条件をみたす。これを定常解 (**定常波**) という。

$\Omega = (0,4)\times(0,4)$ における初期値境界値問題を Fourier の方法で解いた解の公式

$$u(x,y,t) = \sum_{m,n=1}^{\infty} \sin\frac{n\pi x}{4} \sin\frac{m\pi y}{4} (a_{nm}\cos(\frac{\sqrt{n^2+m^2}c\pi t}{4}) + b_{nm}\sin(\frac{\sqrt{n^2+m^2}c\pi t}{4}))$$

である。私の Ω での数値実験では、

$$c = 1, \quad u(x,y,t) = \sin 2\pi x \sin \pi y \cos\sqrt{5}\pi t$$

とした。

初期値は

$$\begin{aligned} \phi(x,y) &= \sin 2\pi x \sin \pi y \\ \psi(x,y) &= 0 \end{aligned}$$

である。

第4章　Javaによるシミュレーションプログラミング

4.1　１次元、２次元方程式のprogram〜WaveAll_2_1.java〜

私は三井さんのプログラムWaveAll_2_0.javaを改良し、いろいろな平面波と定常波をシュミレーションができるWaveAll_2_1.javaを作りました。

三井さんのMitsuiWorldを改良し、Mitsui_and_Itoを作りました。Mitsui_and_Itoとは数学のグラフを描くのに便利なパッケージです。これを使うとJavaの座標を考えずに数学の座標だけを考えてグラフを描くことができます。また等高線を引くことができます。

〜WaveAll_2_1.java〜

```
/**  １次元波動方程式と２次元波動方程式を求めるプログラミングです。
 *   マルチスレッドプログラミングを使っています。
 *   ダブルバッファリングを使っています。
 *
 *   これをコンパイルするにはMitsui_and_Itoが必要です。
 *   MitsuiWorldが使える状態であれば普通のコンパイル javac WaveAll_2_1.java
 *   と普通の起動方法 appletviewer Wave1d_2_1.java
 *   でＯＫ。
 *   ただし appletviewer WaveAll_2_1.java としたければコメント欄のどこかに
 *
 *   <applet code = WaveAll_2_1 width=500 height=500></applet>
 *
 *   というものが入ってないといけない。
 *   それとMitsuiWorldの使い方によっては
 *
 *   import Mitsui.*;
 *
 *   を消すこと。
 *   Mitsui_and_Itoをパッケージとして使わない人はこれを消してください
 *
 */


import java.applet.*;
```

```
import java.awt.*;
import Mitsui.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import java.util.*;

public class WaveAll_2_1 extends Applet
    implements Runnable,ActionListener,ItemListener{

    Thread th = null;                                   //スレッド
    Button b_start,b_stop,b_next,b_back,b_reverse;      //各ボタン
    Button b_1dset,b_2dset;
    Label l_time,l_area;                                //各ラベル
    Label2 l_func;
    Choice c_dimension,c_boundary;     //関数を選ぶ GUI
    boolean runmove=false;             //スレッドが動いてるか止まってるか
    boolean reverseflag=false;         //reverse の時に時間をマイナスにする

    int im_t,re_t;                          //時間の経過
    int wmax,hmax;                          //画面のサイズ

    //もう一つの画面の定義
    Graphics bg;
    Image buf;

    //画面作成に必要なもの
    Mitsui_and_Ito m;
    //2 次元
    double xmin2,xmax2,ymin2,ymax2,zmin2,zmax2;
    int nx2,ny2;
    double dx2,dy2;
    double tau2;
    //1 次元
    double xmin1,xmax1,ymin1,ymax1;
    int nx1;
    double dx1;
    double tau1;

    int n1=100;
    int n2=100;

    double[][] u2_1 = new double[n2+1][n2+1];
    double[][] u2_2 = new double[n2+1][n2+1];
    double[][] u2_3 = new double[n2+1][n2+1];
    double[]   u1_1 = new double[n1+1];
    double[]   u1_2 = new double[n1+1];
    double[]   u1_3 = new double[n1+1];
```

```
int nfunc1=0;                          //１次元の関数を変える
int nfunc2=0;                          //２次元の関数を変える
int dimension=0;                       //次元を変える/0=１次元,1=２次元
int boundary=0;                       //境界条件を変える

//視覚を変える変数
int lx,ly,lz;                            //各軸の長さ
int angx,angy;                           //視覚の角度
int x0,y0;                               //原点の位置

//初期設定
public void init(){
    setLayout(new BorderLayout());
    Panel p  = new Panel();           //NORTH 地区を枠決めしてる
    p.setLayout(new GridLayout(3,1,0,0));
    Panel p1 = new Panel();           //１段目
    Panel p2 = new Panel();           //２段目
    Panel p3 = new Panel();           //３段目


    //start ボタンの設置
    b_start=new Button("START");
    b_start.addActionListener(this);
    p2.add(b_start);

    //stop ボタンの設置
    b_stop=new Button("STOP");
    b_stop.addActionListener(this);
    p2.add(b_stop);

    //reverse ボタンの設置
    b_reverse=new Button("REVERSE");
    b_reverse.addActionListener(this);
    p2.add(b_reverse);

    //back ボタンの設置
    b_back=new Button("BACK");
    b_back.addActionListener(this);
    p2.add(b_back);

    //next ボタンの設置
    b_next=new Button("NEXT");
    b_next.addActionListener(this);
    p2.add(b_next);

    //dimension チョイスの設置
    c_dimension=new Choice();
    c_dimension.addItem("１次元波動方程式");
    c_dimension.addItem("２次元波動方程式");
```

```
        c_dimension.addItem("２次元（等高線）");
        c_dimension.addItemListener(this);
        p1.add(c_dimension);

        //boundary チョイスの設置
        c_boundary=new Choice();
        c_boundary.addItem("Dirichlet");
        c_boundary.addItem("Neumann(素朴)");
        c_boundary.addItem("Neumann(仮想格子点)");
        c_boundary.addItemListener(this);
        p1.add(c_boundary);

        //１次元のセッティングを行うボタン
        b_1dset = new Button("１次元の設定");
        b_1dset.addActionListener(this);
        p1.add(b_1dset);

        //２次元のセッティングを行うボタン
        b_2dset = new Button("２次元の設定");
        b_2dset.addActionListener(this);
        p1.add(b_2dset);

        //ラベル l_time の設置　時間を表示する。
        l_time = new Label("           ");
        p2.add(l_time);

        //ラベル l_area の設置　描写範囲の表示
        l_area = new Label("                                                    ");
        p3.add(l_area);

        //ラベル l_func の設置　選んだ関数の表示
        l_func = new Label2(
         "φ=sin(π x)                                                    ",

"ψ=0                                                                    ");

        p3.add(l_func);

        p.add(p1);
        p.add(p2);
        p.add(p3);
        add(p,"North");

        //画面の横と縦の長さを得る。
        wmax = getSize().width;
        hmax = getSize().height;

        //描写範囲を決める
        //２次元 [xmin2,xmax2] × [ymin2,ymax2] × [zmin2,zmax2]
```

```
    xmin2 = 0.0;
    xmax2 = 4.0;
    ymin2 = 0.0;
    ymax2 = 4.0;
    zmin2 = -1.0;
    zmax2 = 1.0;

    //1次元 [xmin1,xmax1] × [ymin1,ymax1]
    xmin1 = -0.2;
    xmax1 = 1.2;
    ymin1 = -1.2;
    ymax1 = 1.8;

    //作図用の描写画面の作成
    buf = createImage(wmax,hmax);
    bg = buf.getGraphics();

    //Mitsui_and_Ito
    m = new Mitsui_and_Ito(); // mk
    m.setGraphics(bg);
    m.setScreenSize(wmax,hmax);

    //分割数
    nx1 = 100;              //1次元のx分割数
    nx2 = 40;               //2次元のxの分割数大きすぎると動作が遅い
    ny2 = 40;               //2次元のyの分割数大きすぎると動作が遅い

    //時刻間隔
    tau1 = 0.01;
    tau2 = 0.07;

    //視覚を変える変数の初期値
    lx=0;
    ly=0;
    lz=0;
    x0=y0=0;
    angx=angy=0;
}

//起動と同時にスレッドを走らせる
public void start(){
    if(th==null){
        th=new Thread(this);
        th.start();
    }
}
private void draw_contour() {
    for (double h=-1.0;h<=1.0;h+=0.1) {
        if (h < 0.0)
```

```
            m.setColor(Color.blue);
        else
            m.setColor(Color.red);
        l_time.setText("t="+re_t*tau2);
        double[][] u = f2(im_t);
        m.contln(xmin2, xmax2, ymin2, ymax2, u, nx2, ny2, h);
    }
}
//スレッドの実装
public void run(){
    while(true){                    //起動中スレッドを走らせる。
        while(!runmove){            //start ボタンを押すことによって
        }                           //ここからぬけ、スレッドが有効になる。
        if(!reverseflag)
            re_t++;
        else
            re_t--;

        im_t++;

        bg.clearRect(0,0,wmax,hmax);        //画面をクリア
        if(dimension==0){                   //１次元波動方程式のとき
            l_time.setText("t="+re_t*tau1);
            double[] u = f1(im_t);
            m.setColor(Color.black);
            drawAxis();
            m.setColor(Color.pink);
            m.move(0.0,u[0]);
            for(int i=1;i<=nx1;i++)
                m.draw(i*dx1,u[i]);
        }
        if(dimension==1){                       //２次元波動方程式のとき
            l_time.setText("t="+re_t*tau2);
            double[][] u = f2(im_t);            //計算した結果を格納する
            m.hideBirdView(u,nx2,ny2,1);        //メモリ内で描く
        }
        if (dimension==2) {
            // 等高線描画
            draw_contour();
            /*
            for (double h=-1.0;h<=1.0;h+=0.1) {
                if (h < 0.0)
                    m.setColor(Color.blue);
                else
                    m.setColor(Color.red);
                l_time.setText("t="+re_t*tau2);
                double[][] u = f2(im_t);
                    m.contln(xmin2, xmax2, ymin2, ymax2, u, nx2, ny2, h);
                }
```

```
            }
            */
        }
        repaint();
        try{
            Thread.sleep(100);          //100 ミリ秒ストップ
        }catch(InterruptedException e){}
    }
}


//終了と同時にスレッドも終了
public void stop(){
    if(th!=null){
        th = null;
    }
}

public void actionPerformed(ActionEvent e){
    //start ボタンを押したときのアクション
    if(e.getSource()==b_start){
        runmove=true;                //再びグラフを描き始める
    }

    //stop ボタンを押したときのアクション
    if(e.getSource()==b_stop){
        if(!runmove){                              //一時停止のとき
            reverseflag=false;                     //初期化
            im_t=0;
            re_t=0;
            repaint();
        }
        else{                                     //これで一時停止
            runmove=false;
        }
    }

    //back ボタンを押したときのアクション
    if(e.getSource()==b_back){
        if(!reverseflag)
            re_t--;
        else
            re_t++;

        if(!runmove){
            if(dimension==0){
                for(int i=0;i<=nx1;i++){
                    u1_2[i] = u1_1[i];
                    u1_1[i] = u1_3[i];
```

```
            }
            double[] u = f1(im_t);
            for(int i=0;i<=nx1;i++){
                u1_2[i] = u1_1[i];
                u1_1[i] = u1_3[i];
                u1_3[i] = u1_2[i];
            }

            l_time.setText("t="+(double)re_t*tau1);
            bg.clearRect(0,0,wmax,hmax);        //画面をクリア
            m.setColor(Color.black);
            drawAxis();
            m.setColor(Color.pink);
            m.move(0.0,u1_3[0]);
            for(int i=1;i<=nx1;i++)
                m.draw(i*dx1,u1_3[i]);
        }
        if(dimension==1 || dimension==2){
            for(int i=0;i<=nx2;i++){
                for(int j=0;j<=ny2;j++){
                    u2_2[i][j] = u2_1[i][j];
                    u2_1[i][j] = u2_3[i][j];
                }
            }
            double[][] u = f2(im_t);
            for(int i=0;i<=nx2;i++){
                for(int j=0;j<=ny2;j++){
                    u2_2[i][j] = u2_1[i][j];
                    u2_1[i][j] = u2_3[i][j];
                    u2_3[i][j] = u2_2[i][j];
                }
            }

            l_time.setText("t="+(double)re_t*tau2);
            bg.clearRect(0,0,wmax,hmax);        //画面をクリア
            m.hideBirdView(u2_3,nx2,ny2,1);       //メモリ内で描く
        }
        repaint();
    }
}

//next ボタンを押したときのアクション
if(e.getSource()==b_next){
    if(!runmove){                       //一時停止状態のときのみ
        if(!reverseflag)
        re_t++;
    else
        re_t--;                          //次の時間のグラフを見れる。
```

```
        bg.clearRect(0,0,wmax,hmax);       //画面をクリア
        if(dimension==0){
            l_time.setText("t="+(double)re_t*tau1);
            double[] u = f1(im_t);
            m.setColor(Color.black);
            drawAxis();
            m.setColor(Color.pink);
            m.move(0.0,u[0]);
            for(int i=1;i<=nx1;i++)
                m.draw(i*dx1,u[i]);
        }
        if(dimension==1){
            l_time.setText("t="+(double)re_t*tau2);
            double[][] u = f2(im_t);      //計算した結果を格納する
            m.hideBirdView(u,nx2,ny2,1);   //メモリ内で描く
        }
        if (dimension == 2) {
            draw_contour();
            /*
              for (double h=-1.0;h<=1.0;h+=0.1) {
              if (h < 0.0)
              m.setColor(Color.blue);
              else
              m.setColor(Color.red);

            l_time.setText("t="+(double)re_t*tau2);
            double[][] u = f2(im_t);
            m.contln(xmin2, xmax2, ymin2, ymax2, u, nx2, ny2, h);
            */
        }
        repaint();
    }
}

//reverse ボタンを押したときのアクション
if(e.getSource()==b_reverse){
    //逆流してるかしてないか。
    if(reverseflag)
        reverseflag=false;
    if(!reverseflag)
        reverseflag=true;

    if(dimension==0){
        double[]u = f1(im_t);
        for(int i=0;i<=nx1;i++){
            u1_2[i] = u1_1[i];
            u1_1[i] = u1_3[i];
            u1_3[i] = u1_2[i];
        }
```

```
            }
            if(dimension==1 || dimension == 2){
                double[][] u = f2(im_t);
                for(int i=0;i<=nx2;i++){
                    for(int j=0;j<=ny2;j++){
                        u2_2[i][j]=u2_1[i][j];
                        u2_1[i][j]=u2_3[i][j];
                        u2_3[i][j]=u2_2[i][j];
                    }
                }
            }
        }

        //１次元波動方程式の設定をするダイアログを作っている。
        if(e.getSource()==b_1dset){
            String s = ("初期条件を選んでください");

            //初期値選択欄の作成
            Choice c_func1 = new Choice();
            c_func1.addItem("固有振動");
            c_func1.addItem("別れては重なる波");
            c_func1.addItem("右方向に動く波");
            c_func1.addItem("左方向に動く波");
            c_func1.addItem("二つの波の合体");

            //分割数記入欄の作成
            NumericField inputN = new NumericField(""+nx1);
            inputN.setBorder
                (new TitledBorder("分割数を指定してください (=<100)"));

            //時間刻み記入欄の作成
            NumericField  inputTau = new NumericField(""+tau1);
            inputTau.setBorder
                (new TitledBorder("時間刻みを指定してください"));

            //描写範囲指定欄の作成
            NumericField  inputArea =
                new NumericField(xmin1+" "+xmax1+"         "+ymin1+" "+ymax1);
            inputArea.setBorder
                (new TitledBorder("描写範囲を指定してください　数の間には空白を入れ
て下さい。"));

            //オブジェクトを一つにまとめて
            Object[] obj = {s,c_func1,inputN,inputTau,inputArea};

            //ダイアログの作成
            int ans = JOptionPane.showConfirmDialog(this,obj,
                                                "１次元波動方程式の設定",
                                                 JOptionPane.OK_CANCEL_OPTION);
```

```
if(ans==0){                              //OK を押した時
    nfunc1  = c_func1.getSelectedIndex();

    if(dimension==0){
        //式の記述
        switch(nfunc1){
        case 0:
            l_func.setText("φ=sin(π x)","ψ=0");
            break;
        case 1:
            l_func.setText("φ=(sin(π (10x-0.5))+1)/4",
                           "ψ=0");
            break;
        case 2:
            l_func.setText("φ=(sin(π (10x-0.5))+1)/4",
                           "ψ=-2.5 π cos(π (10x-0.5))");
            break;
        case 3:
            l_func.setText("φ=(sin(π (10x-0.5))+1)/4",
                           "ψ=2.5 π cos(π (10x-0.5))");
            break;
        case 4:
            l_func.setText("φ 1=(sin(π (10(x+0.4)-0.5))+1)/4",
                           "φ 2=(sin(π (10(x-0.4)-0.5))+1)/4");
            break;
        default:
            l_func.setText("未定","未定");
            break;
        }
    }

    nx1 = Integer.parseInt(inputN.getText().trim());
    tau1 = Double.valueOf(inputTau.getText().trim()).doubleValue();

    //描写範囲を読み込む時は「分割」をつかう。
    String str = inputArea.getText().trim();
    //空白区切りで分割する
    StringTokenizer st = new StringTokenizer(str," ");
    if(st.countTokens()==4){              //分割数が４のとき
        xmin1 = Double.valueOf(st.nextToken()).doubleValue();
        xmax1 = Double.valueOf(st.nextToken()).doubleValue();
        ymin1 = Double.valueOf(st.nextToken()).doubleValue();
        ymax1 = Double.valueOf(st.nextToken()).doubleValue();
    }


    runmove  = false;                        //初期状態に戻す
    reverseflag=false;
```

```
        im_t=0;
        re_t=0;
        repaint();
    }
}

if(e.getSource()==b_2dset){
    String s = ("初期条件を選んでください");

    Choice c_func2 = new Choice();
    c_func2.addItem("φ=sin(π x)+sin(π y), ψ=0");
    c_func2.addItem("定常波");
    c_func2.addItem("x 軸に平行な平面波");
    c_func2.addItem("y 軸に平行な平面波");
    c_func2.addItem("平面波の合体");
    c_func2.addItem("斜め方向の平面波");

    //ダイアログは持ち込むオブジェクトを縦に並べる。しかし
    //パネルを使うことにより横にもコンテンツを増やした。
    Panel pdivide = new Panel();
    JLabel N = new JLabel("分割数");
    NumericField inputNx = new NumericField(""+nx2,10);
    NumericField inputNy = new NumericField(""+ny2,10);
    inputNx.setBorder(new TitledBorder("Nx(=<50)"));
    inputNy.setBorder(new TitledBorder("Ny(=<50)"));
    pdivide.add(N);
    pdivide.add(inputNx);
    pdivide.add(inputNy);

    NumericField  inputTau = new NumericField(""+tau2);
    inputTau.setBorder
        (new TitledBorder("時間刻みを指定してください"));

    NumericField  inputArea =
        new NumericField(xmin2+" "+xmax2+"         "+
                         ymin2+" "+ymax2+"         "+zmin2+" "+zmax2);
    inputArea.setBorder
        (new TitledBorder("描写範囲の指定　数の間には空白を入れて下さい。"));
    //角度を変える欄の作成
    Panel pangle = new Panel();
    JLabel A = new JLabel("角度変え（ラジアン）");
    NumericField inputAngx = new NumericField(""+angx,10);
    NumericField inputAngy = new NumericField(""+angy,10);
    inputAngx.setBorder(new TitledBorder("水平方向"));
    inputAngy.setBorder(new TitledBorder("垂直方向"));
    pangle.add(A);
    pangle.add(inputAngx);
    pangle.add(inputAngy);
```

```
//原点移動の欄の作成
Panel porigin = new Panel();
JLabel O = new JLabel("原点移動（ピクセル）");
NumericField inputX0 = new NumericField(""+x0,10);
NumericField inputY0 = new NumericField(""+y0,10);
inputX0.setBorder(new TitledBorder("右向き"));
inputY0.setBorder(new TitledBorder("上向き"));
porigin.add(O);
porigin.add(inputX0);
porigin.add(inputY0);

//軸の長さ変更欄の作成
Panel plength = new Panel();
JLabel L = new JLabel("サイズ変え");
NumericField inputLx = new NumericField(""+lx,8);
NumericField inputLy = new NumericField(""+ly,8);
NumericField inputLz = new NumericField(""+lz,8);
inputLx.setBorder(new TitledBorder("ｘ軸"));
inputLy.setBorder(new TitledBorder("ｙ軸"));
inputLz.setBorder(new TitledBorder("ｚ軸"));
plength.add(L);
plength.add(inputLx);
plength.add(inputLy);
plength.add(inputLz);

Object[] obj = {s,c_func2,pdivide,inputTau,inputArea,pangle,
                porigin,plength};

int ans = JOptionPane.showConfirmDialog(this,obj,
                                    "２次元波動方程式の設定",
                                     JOptionPane.OK_CANCEL_OPTION);
if(ans==0){
    nfunc2  = c_func2.getSelectedIndex();

    if(dimension==1 || dimension == 2){
        //式の記述
        switch(nfunc2){
        case 0:
            l_func.setText("φ=sin(π x)+sin(π y)",
                           "ψ=0");
            break;
        case 1:
            l_func.setText( "φ=(sin2 π x)(sin π y)",
                            "ψ=0");
            break;
        case 2:
            l_func.setText("φ=(sin(π (2x-0.5))+1)/4",
                           "ψ=-π cos(π (2x-0.5))/2");
            break;
```

```
                    case 3:
                        l_func.setText("φ=(sin(π (2y-0.5))+1)/4",
                                       "ψ=-π cos(π (2y-0.5))/2");
                        break;
                    case 4:
                        l_func.setText("φ 1=(sin(π (2(x+1)-0.5))+1)/4",
                                       "φ 2=(sin(π (2(x-2)-0.5))+1)/4");
                        break;
                    case 5:
                         l_func.setText("φ=(sin(π (2(1/sqrt(2.0)(x+y))-0.5)+1)/4",
ψ=-π cos(π (2(1/sqrt(2.0)(x+y))-0.5))/2" );

                        break;
                    default:
                        l_func.setText("未定","未定");
                        break;
                    }
                }

                nx2 = Integer.parseInt(inputNx.getText().trim());
                ny2 = Integer.parseInt(inputNy.getText().trim());

                tau2 = Double.valueOf(inputTau.getText().trim()).doubleValue();

                String str = inputArea.getText().trim();
                StringTokenizer st = new StringTokenizer(str," ");
                if(st.countTokens()==6){
                    xmin2 = Double.valueOf(st.nextToken()).doubleValue();
                    xmax2 = Double.valueOf(st.nextToken()).doubleValue();
                    ymin2 = Double.valueOf(st.nextToken()).doubleValue();
                    ymax2 = Double.valueOf(st.nextToken()).doubleValue();
                    zmin2 = Double.valueOf(st.nextToken()).doubleValue();
                    zmax2 = Double.valueOf(st.nextToken()).doubleValue();
                }

                angx = Integer.parseInt(inputAngx.getText().trim());
                angy = Integer.parseInt(inputAngy.getText().trim());

                x0 = Integer.parseInt(inputX0.getText().trim());
                y0 = Integer.parseInt(inputY0.getText().trim());

                lx = Integer.parseInt(inputLx.getText().trim());
                ly = Integer.parseInt(inputLy.getText().trim());
                lz = Integer.parseInt(inputLz.getText().trim());

                runmove  = false;                          //初期状態に戻す
                reverseflag=false;
                im_t=0;
                re_t=0;
```

```
            repaint();
        }
    }
}

//dimension,boundary,fund チョイスを選んだとき
public void itemStateChanged(ItemEvent e){
    dimension=c_dimension.getSelectedIndex();  //次元番号を変える
    boundary =c_boundary.getSelectedIndex();   //境界条件番号を変える
    runmove  = false;                          //初期状態に戻す
    reverseflag=false;
    im_t=0;
    re_t=0;
    repaint();
}

public void paint(Graphics g){
    if(im_t==0){
        bg.clearRect(0,0,wmax,hmax);
        if(dimension==0){
            dx1  = 1.0/nx1;              //１次元のｘ格子点
            l_time.setText("t=0.0");     //時刻と描写範囲の表示
            l_area.setText("["+xmin1+","+xmax1+"] × ["+ymin1+","+ymax1+"]");
            m.setArea(xmin1,xmax1,ymin1,ymax1);

            double[] u = f1(im_t);
            m.setColor(Color.black);
            drawAxis();
            m.setColor(Color.pink);
            m.move(0.0,u[0]);
            for(int i=1;i<=nx1;i++)
                m.draw(i*dx1,u[i]);
        }
        if(dimension==1 || dimension==2){
            dx2 = (xmax2-xmin2)/nx2;     //２次元のｘ格子点
            dy2 = (ymax2-ymin2)/ny2;     //２次元のｙ格子点
            l_time.setText("t=0.0");     //時刻と描写範囲の表示
            l_area.setText("["+xmin2+","+xmax2+"] × ["+ymin2+","+ymax2+
                           "] × ["+zmin2+","+zmax2+"]");
            //視覚を変えている
            m.changeAngle(angx,angy);
            m.changePosition(x0,y0);
            m.changeSize(lx,ly,lz);

            //描き始める。
            m.setArea2(xmin2,xmax2,ymin2,ymax2,zmin2,zmax2);
            double[][] u = f2(im_t);
            m.setColor(Color.pink);
            m.hideBirdView(u,nx2,ny2,1);
```

```
        }
    }
    g.drawImage(buf,0,0,this);            //もう一つの画面をくっつける
}

public double[][] f2(int t){
    double lambdax = tau2 / dx2;
    double lambday = tau2 / dy2;
    double lambdax2 = lambdax * lambdax;
    double lambday2 = lambday * lambday;

    if(t==0){                                    //t=0 のとき
        for(int i=0;i<=nx2;i++)
            for(int j=0;j<=ny2;j++)
                u2_1[i][j] = phi(xmin2+i*dx2,ymin2+j*dy2);   //初期値代入

        return u2_1;
    }

    if(t==1){
        for(int i=1;i<nx2;i++)
            for(int j=1;j<ny2;j++)
                u2_2[i][j] = (1.0-lambdax2-lambday2) * u2_1[i][j] +
                    0.5 * lambdax2 * (u2_1[i-1][j]+u2_1[i+1][j]) +
                    0.5 * lambday2 * (u2_1[i][j-1]+u2_1[i][j+1]) +
                    tau2 * psi(xmin2+i*dx2,ymin2+j*dy2);

        if(boundary==0){                          //Dirichlet 境界条件
            for(int i=0;i<=nx2;i++)
                u2_2[i][0]=u2_2[i][ny2]=0.0;
            for(int i=0;i<=ny2;i++)
                u2_2[0][i]=u2_2[nx2][i]=0.0;
        }
        if(boundary==1){                          //Neumann 境界条件
            for(int i=0;i<=nx2;i++){
                u2_2[i][0]=u2_2[i][1];
                u2_2[i][ny2]=u2_2[i][ny2-1];
            }
            for(int i=0;i<=ny2;i++){
                u2_2[0][i]=u2_2[1][i];
                u2_2[nx2][i]=u2_2[nx2-1][i];
            }
        }
        if(boundary==2){                          //仮想格子点
            for(int i=1;i<nx2;i++){
                u2_2[i][0]=(1.0-lambdax2-lambday2) * u2_1[i][0] +
                    0.5 * lambdax2 * (u2_1[i-1][0]+u2_1[i+1][0]) +
                    0.5 * lambday2 * (2*u2_1[i][1]) +
                    tau2 * psi(xmin2+i*dx2,ymin2);
```

```
            u2_2[i][ny2]=(1.0-lambdax2-lambday2) * u2_1[i][ny2] +
                0.5 * lambdax2 * (u2_1[i-1][ny2]+u2_1[i+1][ny2]) +
                0.5 * lambday2 * (2*u2_1[i][ny2-1]) +
                tau2 * psi(xmin2+i*dx2,ymin2+ny2*dy2);
        }
        for(int j=1;j<ny2;j++){
            u2_2[0][j]=(1.0-lambdax2-lambday2) * u2_1[0][j] +
                0.5 * lambdax2 * (2*u2_1[1][j]) +
                0.5 * lambday2 * (u2_1[0][j-1]+u2_1[0][j+1]) +
                tau2 * psi(xmin2,ymin2+j*dy2);
            u2_2[nx2][j]=(1.0-lambdax2-lambday2) * u2_1[nx2][j] +
                0.5 * lambdax2 * (2*u2_1[nx2-1][j]) +
                0.5 * lambday2 * (u2_1[nx2][j-1]+u2_1[nx2][j+1]) +
                tau2 * psi(xmin2+nx2*dx2,ymin2+j*dy2);
        }
        u2_2[0][0]=(1.0-lambdax2-lambday2) * u2_1[0][0] +
                0.5 * lambdax2 * (2*u2_1[1][0]) +
                0.5 * lambday2 * (2*u2_1[0][1]) +
                tau2 * psi(xmin2,ymin2);
        u2_2[nx2][0]=(1.0-lambdax2-lambday2) * u2_1[nx2][0] +
                0.5 * lambdax2 * (2*u2_1[nx2-1][0]) +
                0.5 * lambday2 * (2*u2_1[nx2][1]) +
                tau2 * psi(xmin2+nx2*dx2,ymin2);
        u2_2[0][ny2]=(1.0-lambdax2-lambday2) * u2_1[0][ny2] +
                0.5 * lambdax2 * (2*u2_1[1][ny2]) +
                0.5 * lambday2 * (u2_1[0][ny2-1]) +
                tau2 * psi(xmin2,ymin2+ny2*dy2);
        u2_2[nx2][ny2]=(1.0-lambdax2-lambday2) * u2_1[nx2][ny2] +
                0.5 * lambdax2 * (2*u2_1[nx2-1][ny2]) +
                0.5 * lambday2 * (2*u2_1[nx2][ny2-1]) +
                tau2 * psi(xmin2+nx2*dx2,ymin2+ny2*dy2);

    }
    return u2_2;
}

else{                                       //t>=2 のとき
    for(int i=1;i<nx2;i++)
        for(int j=1;j<ny2;j++)
            u2_3[i][j] = 2 * (1.0-lambdax2 -lambday2) * u2_2[i][j] +
                lambdax2 * (u2_2[i-1][j]+u2_2[i+1][j]) +
                lambday2 * (u2_2[i][j-1]+u2_2[i][j+1]) - u2_1[i][j];

    if(boundary==0){                         //Dirichlet 境界条件
        for(int i=0;i<=nx2;i++)
            u2_3[i][0]=u2_3[i][ny2]=0.0;
        for(int i=0;i<=ny2;i++)
            u2_3[0][i]=u2_3[nx2][i]=0.0;
    }
```

```
if(boundary==1){                          //Neumann 境界条件
    for(int i=0;i<=nx2;i++){
        u2_3[i][0] = u2_3[i][1] ;
        u2_3[i][ny2]= u2_3[i][ny2-1];
    }
    for(int i=0;i<=ny2;i++){
        u2_3[0][i] = u2_3[1][i];
        u2_3[nx2][i]= u2_3[nx2-1][i];
    }
}
if(boundary==2){                          //仮想格子点
    for(int i=1;i<nx2;i++){
        u2_3[i][0] = 2 * (1.0-lambdax2 -lambday2) * u2_2[i][0] +
            lambdax2 * (u2_2[i-1][0]+u2_2[i+1][0]) +
            lambday2 * (2*u2_2[i][1]) - u2_1[i][0];
        u2_3[i][ny2]=2 * (1.0-lambdax2 -lambday2) * u2_2[i][ny2] +
            lambdax2 * (u2_2[i-1][ny2]+u2_2[i+1][ny2]) +
            lambday2 * (2*u2_2[i][ny2-1]) - u2_1[i][ny2];
    }
    for(int j=1;j<ny2;j++){
        u2_3[0][j] = 2 * (1.0-lambdax2 -lambday2) * u2_2[0][j] +
            lambdax2 * (2*u2_2[1][j]) +
            lambday2 * (u2_2[0][j-1]+u2_2[0][j+1]) - u2_1[0][j];
        u2_3[nx2][j]=2 * (1.0-lambdax2 -lambday2) * u2_2[nx2][j] +
            lambdax2 * (2*u2_2[nx2-1][j]) +
            lambday2 *(u2_2[nx2][j-1]+u2_2[nx2][j+1])-u2_1[nx2][j];
    }
    u2_3[0][0] = 2 * (1.0-lambdax2 -lambday2) * u2_2[0][0] +
            lambdax2 * (2*u2_2[1][0]) +
            lambday2 * (2*u2_2[0][1]) - u2_1[0][0];
    u2_3[nx2][0]= 2 * (1.0-lambdax2 -lambday2) * u2_2[nx2][0] +
            lambdax2 * (2*u2_2[nx2-1][0]) +
            lambday2 * (2*u2_2[nx2][1]) - u2_1[nx2][0];
    u2_3[0][ny2]= 2 * (1.0-lambdax2 -lambday2) * u2_2[0][ny2] +
            lambdax2 * (2*u2_2[1][ny2]) +
            lambday2 * (2*u2_2[0][ny2-1]) - u2_1[0][ny2];
    u2_3[nx2][ny2]=2 * (1.0-lambdax2 -lambday2) * u2_2[nx2][ny2] +
            lambdax2 * (2*u2_2[nx2-1][ny2]) +
            lambday2 * (2*u2_2[nx2][ny2-1]) - u2_1[nx2][ny2];
}

//u1 に u2 の値を u2 に u3 の値を代入
for(int i=0;i<=nx2;i++){
    for(int j=0;j<=ny2;j++){
        u2_1[i][j] = u2_2[i][j];
        u2_2[i][j] = u2_3[i][j];
    }
}
```

```
        return u2_3;
    }

}

public double[] f1(int t){
    double lambda = tau1/dx1;
    double lambda2 = lambda * lambda;

    if(t==0){
        for(int i=0;i<=nx1;i++)
            u1_1[i] = phi(i*dx1);
        return u1_1;
    }

    if(t==1){
        for(int i=1;i<nx1;i++)
            u1_2[i] = (1-lambda2) * u1_1[i] +
                0.5 * lambda2 * (u1_1[i-1]+u1_1[i+1]) + tau1 * psi(i*dx1);

        if(boundary==0){                        //Dirichlet 境界条件
            u1_2[0]=u1_2[nx1]=0;
        }
        if(boundary==1){                        //Neumann 境界条件
            u1_2[0]=u1_2[1];
            u1_2[nx1]=u1_2[nx1-1];
        }
        if(boundary==2){                        //仮想格子点
            u1_2[0] = (1-lambda2) * u1_1[0] +
                0.5 * lambda2 * (2*u1_1[1]) + tau1 * psi(0.0);
            u1_2[nx1] = (1-lambda2) * u1_1[nx1] +
                0.5 * lambda2 * (2*u1_1[nx1-1]) + tau1 * psi(nx1*dx1);
        }

        return u1_2;
    }

    else{
        for(int i=1;i<nx1;i++)
            u1_3[i] = 2 * (1.0-lambda2) * u1_2[i] +
                lambda2 * (u1_2[i-1]+u1_2[i+1]) - u1_1[i];

        if(boundary==0){                       //Dirichlet 境界条件
            u1_3[0] = u1_3[nx1] = 0;
        }
        if(boundary==1){                       //Neumann 境界条件
            u1_3[0]=u1_3[1];
            u1_3[nx1]=u1_3[nx1-1];
        }
```

```
        if(boundary==2){                    //仮想格子点
            u1_3[0] = 2 * (1.0-lambda2) * u1_2[0] +
                lambda2 * (2*u1_2[1]) - u1_1[0];
            u1_3[nx1] = 2 * (1.0-lambda2) * u1_2[nx1] +
                lambda2 * (2*u1_2[nx1-1]) - u1_1[nx1];
        }

        //u1 に u2 の値を u2 に u3 の値を代入
        for(int i=0;i<nx1;i++){
            u1_1[i] = u1_2[i];
            u1_2[i] = u1_3[i];
        }

        return u1_3;
    }
}

static double U(double x) {
    if (1 <= x && x <= 2)
        return (Math.sin(Math.PI*(x*2-0.5))+1)/4;
    else
        return 0.0;
}
static double dUdx(double x) {
    if (1 <= x && x <= 2 )
        return -2*Math.PI*Math.cos(Math.PI*(x*2-0.5))/4;
    else
        return 0.0;
}
//初期値φ
public double phi(double x,double y){
    double cx = (xmax2+xmin2)/2;
    double cy = (ymax2+ymin2)/2;
    double r  = (xmax2-xmin2)/5;

    if (nfunc2==0){
        return Math.sin(Math.PI*x)/2+Math.sin(Math.PI*y)/2;
    }
    if (nfunc2==1){
        return (Math.sin(Math.PI*2*x)*Math.sin(Math.PI*y))/2;
    }
    if (nfunc2==2) {
        return U(x);
    }
    if (nfunc2==3){
        return U(y);
    }
    if (nfunc2 == 4){
        if(0 <=x && x <= 1)
```

```
            return U(x+1);
        else if (3<=x && x<=4)
            return U(x-2);
        else
            return 0.0;
    }
    if (nfunc2== 5){
             return U(1/Math.sqrt(2.0)*(x+y));
    }
    else
        return 0.0;
}

//初期条件ψ
public double psi(double x,double y){
    double cx = (xmax2+xmin2)/2;
    double cy = (ymax2+ymin2)/2;
    double r  = (xmax2-xmin2)/5;

    if(nfunc2==2){
        return dUdx(x);
    }
    if(nfunc2==3){
        return dUdx(y);
    }
    if(nfunc2 == 4){
        if(0 <=x && x <= 1)
            return dUdx(x+1);
        else if (3 <=x && x<= 4)
            return - dUdx(x-2);
        else
            return 0.0;
    }
    if(nfunc2 == 5){
            return dUdx(1/Math.sqrt(2.0)*(x+y));
    }

    else
        return 0.0;
}


static double U1(double x) {
    if (0.4 <= x && x <= 0.6)
        return (Math.sin(Math.PI*(10*x-0.5))+1)/4;
    else
        return 0.0;
}
static double dU1dx(double x) {
```

```
    if (0.4 <= x && x <= 0.6)
        return -2.5*Math.PI*Math.cos(Math.PI*(10*x-0.5));
    else
        return 0.0;

}


//初期値φ
public double phi(double x){
    if(nfunc1 == 0){
        return Math.sin(Math.PI*x);
    }
    if(nfunc1 == 1){
            return U1(x);
    }
    if(nfunc1 == 2){
            return U1(x);
    }
     if(nfunc1 == 3){
            return U1(x);
    }
     if(nfunc1 == 4){
        if(0.0<=x && x<=0.2)
            return U1(x+0.4);
        else if(0.8<=x && x<=1.0)
            return U1(x-0.4);
        else
            return 0.0;
    }
    else
        return 0.0;
}

//初期条件ψ
public double psi(double x){
    if(nfunc1 == 0){
        return 0.0;
    }
    if(nfunc1 == 1){
        return 0.0;
    }
    if(nfunc1 == 2){
        return dU1dx(x);
    }
    if(nfunc1 == 3){
        return - dU1dx(x);
    }
    if(nfunc1 == 4){
```

```
        if(0.0<=x && x<=0.2)
            return  dU1dx(x+0.4);
        else if(0.8<=x && x<=1.0)
            return - dU1dx(x-0.4);
        else
            return 0.0;
    }
    else
        return 0.0;
}

public void drawAxis(){
    m.move(0.0,0.0); m.draw(1.025,0.0);
    for(double i=0.0;i<=1.0;i+=0.1){
        m.move(i,0.025); m.draw(i,-0.025);
    }
    m.move(0.0,1.05); m.draw(0.0,-1.05);
    for(double i=1.0;i>=-1;i-=0.2){
        m.move(-0.01,i); m.draw(0.01,i);
    }
}

//数字しか入らないテキストフィールドの作成
class NumericField extends JTextField{
    public NumericField(String begin){
        super(begin);
        enableEvents(AWTEvent.KEY_EVENT_MASK);
    }

    public NumericField(String begin,int num){
        super(begin,num);
        enableEvents(AWTEvent.KEY_EVENT_MASK);
    }

    protected void processKeyEvent(KeyEvent e){
        String validValues = "0123456789.- ";
        int code = e.getKeyCode();
        switch(code){
        case 39:break;                          //右カーソル
        case 37:break;                          //左カーソル
        default:
            char chr = e.getKeyChar();
            if(chr >= ' ' && validValues.indexOf( chr )==-1){
                return;
            }
            break;
        }
        super.processKeyEvent(e);
    }
```

```
    }

    //２行にまたがるラベル。
    class Label2 extends JComponent{
        JLabel row1,row2;

        public Label2(String s1,String s2){
            setLayout(new BoxLayout(this,BoxLayout.Y_AXIS));
            row1 = new JLabel(s1);
            row2 = new JLabel(s2);
            add(row1);
            add(row2);
        }

        public void setText(String s1,String s2){
            row1.setText(s1);
            row2.setText(s2);
        }
    }
}
```

4.2 等高線についてのprogram〜toukousen3.java〜

〜toukousen3.java〜

```
// *  <applet code="toukousen3" width=500 height=500></applet>

import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import Mitsui.*;   //Mitsui_Ito の入っているパッケージ Mitsui を呼び出す。

public class toukousen3 extends Applet {
    Mitsui_and_Ito m = new Mitsui_and_Ito();

    public void init(){
        Graphics g=getGraphics();
        m.setGraphics(g);
        m.setScreenSize(getSize().width,getSize().height);
        m.setColor(Color.red);
        //[a,b]x[c,d]
        m.setArea(-1.2,1.2,-1.2,1.2);

    }

    public void paint(Graphics g) {
        //0, 1, -1, x, y を表示
        g.setColor(Color.red);
        int nx=100,ny=100;
        double xmin=-1.0,xmax=1.0,ymin=-1.0,ymax=1.0;
        double x,y;
        double [][]u=new double[nx+1][ny+1];

        double dx = (xmax-xmin)/nx;
        double dy = (ymax-ymin)/ny;
        for (int i=0;i<=nx;i++ ){
            x = xmin + i * dx;
            for (int j=0;j<=ny;j++ ) {
                y = ymin + j * dy;
                u[i][j] = x*x-y*y;
            }
        }
        // 等高線描画
        for (double h=-1.0;h<=1.0;h+=0.1) {
            if (h < 0.0)
                m.setColor(Color.blue);
            else
                m.setColor(Color.red);
            m.contln(xmin, xmax, ymin, ymax, u, nx, ny, h);
/**
```

m.contln(xmin, xmax, ymin, ymax, u, nx, ny, h); は
Mitsui_and_Ito から下の contln を呼び出している。([xmin,xmax] × [ymin,ymax])

```
  public void contln(double xmin, double xmax, double ymin, double ymax,
   double [][]u, int nx, int ny, double h) {
   int i, j;
     double dx = (xmax-xmin)/nx, dy=(ymax-ymin)/ny;
     double x[] = new double[nx+1];
     double y[] = new double[ny+1];
     for (i=0;i<=nx;i++)
     x[i] = xmin + i * dx;
     for (j=0;j<=ny;j++)
     y[j] = ymin + j * dy;
     for (i=0;i<nx;i++){
     for (j=0;j<ny;j++){
     celldraw(x[i],x[i+1],x[i+1],
                      y[j],y[j],y[j+1],
                      u[i][j],u[i+1][j],u[i+1][j+1],h);
                      celldraw(x[i],x[i],x[i+1],
                      y[j],y[j+1],y[j+1],
                      u[i][j],u[i][j+1],u[i+1][j+1],h);
                      }
                      }
 }
 // 等高線描画ルーチン
 public void celldraw(double x1, double x2, double x3,
 double y1, double y2, double y3,
                      double u1, double u2, double u3,
                      double h){

     // 以下、オリジナルでは new double[3] だったが、
     // 明らかなバグであると思われる。
     double px[] = new double[4];
     double py[] = new double[4];
     int n;
     double r;

     n = 0;

     if ((u1-h)*(u2-h)<=0){
         n++;
         if (u1==h){
             px[n] = x1;
             py[n] = y1;
         } else if(u2==h) {
             px[n] = x2;
             py[n] = y2;
         } else {
             r = (u1 - h) / (u1 - u2);
```

```
                px[n] = (1 - r) * x1 + r * x2;
                py[n] = (1 - r) * y1 + r * y2;
            }
        }
        if ((u2-h)*(u3-h)<=0){
            n++;
            if (u2==h){
                px[n] = x2;
                py[n] = y2;
            } else if(u3==h) {
                px[n] = x3;
                py[n] = y3;
            } else {
                r = (u2 - h) / (u2 - u3);
                px[n] = (1 - r) * x2 + r * x3;
                py[n] = (1 - r) * y2 + r * y3;
            }
        }
        if ((u3-h)*(u1-h)<=0){
            n++;
            if (u3==h){
                px[n] = x3;
                py[n] = y3;
            } else if(u1==h) {
                px[n] = x1;
                py[n] = y1;
            } else {
                r = (u3 - h) / (u3 - u1);
                px[n] = (1 - r) * x3 + r * x1;
                py[n] = (1 - r) * y3 + r * y1;
            }
        }

        if (n==2){
            move(px[1], py[1]);
            draw(px[2], py[2]);
        } else if (n==3) {
            move(px[1], py[1]);
            draw(px[2], py[2]);
            move(px[2], py[2]);
            draw(px[3], py[3]);
            move(px[3], py[3]);
            draw(px[1], py[1]);
            }
        }
*/

        }
    }
```

```
}
```

関連図書

[1] Java による波動方程式の数値解析
三井　康之著　２００１年度卒業研究
お薦め度　MAX　私の卒業研究の基盤です。私が言うのも変ですがとてもすばらしい内容でとても勉強になりましたし参考にさせてもらいました。

[2] 応用解析２ノート
桂田　祐史著　明治大学の講義
お薦め度　５　桂田先生の講義のときの教科書です。私はここから勉強しなおしました。とてもいい勉強になります。

[3] 応用数学
藤田　宏著　放送大学教育振興会
お薦め度　５　波動の基本となることを学びました。

[4] ザ、Java ２－対話的に動くホームページの作成技術ー
戸川　隼人著　サイエンス社
お薦め度　４　Java を初めて始めるにはわかりやすいです。

[5] 熱、波動と微分方程式
俣野　博　神保　道夫著　岩波書店
お薦め度　３　私が見た中で波について詳しく書かれていました。

[6] LATEX ２　美文書作成入門
奥村　晴彦著　技術評論社
お薦め度　１　桂田先生から学んだ LATEX の使い方を確認しました。