シミュレーション概論ノート第9講

13　波動方程式

13.1　全般的注意

今回は波動方程式を扱う。拡散方程式が独立した章ではないのに何故独立した章にしたかという突っ込みは却下。意味はありません。

波動方程式はおそらく受講する学生にとっては最も馴染み深い偏微分方程式であろう。波が減衰することなく伝わる事を表現する方程式である。一般に

$$(\frac{\partial^2}{\partial t^2} - c^2\frac{\partial^2}{\partial x^2})u(x,t) = 0 \tag{1}$$

という形をしている。$v = x - ct$, $w = x + ct$ とおくと$\partial_v = \partial_x - 1/c\partial_t$, $\partial_w = \partial_x + 1/c\partial_t$ と書けるので (1) は

$$\frac{\partial^2 u}{\partial w \partial v} = 0 \tag{2}$$

となる。この解は F, G を任意の関数として

$$u = F(x - ct) + G(x + ct) \tag{3}$$

となっていることは自明であろう。従って速度$\pm c$ で動くフレームに乗ると解は不変であり、パルスの伝播など波動現象を記述する方程式となっている。

13.2　波動方程式の陽解法

以下では $c = 1$ の波動方程式 (1) を区間 $0 \le x \le 1$ で考える。また本節では境界条件を

$$u(x,0) = \phi(x), \quad \frac{\partial u(x,0)}{\partial t} = \psi(x), \quad u(0,t) = u(1,t) = 0 \tag{4}$$

としよう。この固定境界での波動方程式は弦の運動を表していると考えられる。尚、境界条件の影響は次節で論じる。

波動方程式 (1) を差分化すると $u(x_j, t_n) \simeq U_j^n$に対して

$$\frac{U_j^{n+1} - 2U_j^n + U_j^{n-1}}{(\Delta t)^2} = \frac{(U_{j+1}^n - 2U_j^n + U_{j-1}^n)}{(\Delta x)^2} \tag{5}$$

が最低次で成り立つ。但し時間刻み、空間刻みはそれぞれΔt, Δx であり $x_j = j\Delta x$, $t_n = n\Delta t$ である。単純な差分は陽解法に対応する。ここで (5) を

$$U_j^{n+1} = 2U_j^n - U_j^{n-1} + \alpha(U_{j+1}^n - 2U_j^n + U_{j-1}^n) \tag{6}$$

と書き換えよう。ここで$\alpha = (\Delta t/\Delta x)^2$である。このように時間発展の安定性を決めるパラメータαが t と x の同一のべきに従うことから、陽解法は前章の様な深刻な問題をひきおこさない[1]。

[1] 勿論、もっと進んだ取り扱いでは波動方程式特有の難しさが別の所にあることに気がつくであろう。しかし本講義のレベルを逸脱しているのでその点については触れない。

時間について２階の微分を含んでいるので初期条件が２つある。またその採り入れ方も初学者には自明ではない。特に時間の微分の効果は

$$u(x,\Delta t) \simeq u(x,0) + \Delta t \frac{\partial u(x,0)}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 u(x,0)}{\partial t^2} \tag{7}$$

という展開から

$$U_j^1 = \Phi_j + \Delta\Psi_j + \frac{\alpha}{2}(\Phi_{j+1} - 2\Phi_j + \Phi_{j-1}) \tag{8}$$

で決まる。ここで$\Phi_j = \phi(x_j) = u(x_j,0)$, $\Psi_j = \psi(x_j) = \partial_t u(x_j,0)$ である。また境界条件はこの場合 $U_0^n = U_N^n = 0$ である。

波動方程式のアルゴリズムは

- T: 時刻の上限、M: 時間ステップ、N空間格子点の数の設定。

  $\Delta x = 1/N$, $\Delta t = T/M$, $\alpha = (\Delta t/\Delta x)^2$となる。

- 初期設定

  - 空間格子点 $j = 0,1,\cdots,N$の順に

    $$U_j = \Phi_j$$

    を繰り返す。
  - 空間格子点 $j = 0,1,\cdots,N-1$ の順に

    $$UU_j = U_j + \Delta t\Psi_j + \frac{\alpha}{2}(\Phi_{j+1} - 2\Phi_j + \Phi_{j-1})$$

    を繰り返す。
  - 境界条件

    $$UU_0 = UU_N = 0; \quad UUU_0 = UUU_N = 0$$

    を代入する。

- 時間発展 $n = 1,2,\cdots,M-1$ の順に

  - 格子点 $j = 1,2,\cdots,N-1$ の順に

    $$UUU_j = 2UU_j - U_j + \alpha(UU_{j+1} - 2UU_j + UU_{j-1})$$

    を繰り返す。
  - 格子点 $j = 1,2,\cdots,N-1$ の順に

    $$U_j = UU_j; \quad UU_j = UUU_j$$

    を繰り返す。

  を繰り返す。

というものになるであろう。

例によって Fortran と C のプログラムを並べておく。システムとして格子点を 0 から N まで用意していることに注意せよ。ここでは$\phi(x) = u(x,0) = 2x(1-x)$, $\psi(x) = \partial_x u(x,0) = 0$ とし, $\Delta t = 1/50$, $\Delta x = 1/20$ として $t = 1$ までの振舞を計算している。またΔt を固定して$\Delta x = 1/100$ としたらどうなるかを計算してみよ。余裕があったら解析解

$$u(x,t) = \frac{8}{\pi^3}\sum_{n=1}^{\infty}\frac{1}{(2n-1)^3}\{\sin((2n-1)\pi(x+t)) + \sin((2n-1)\pi(x-t))\}$$

と比べたり、システムサイズや時間刻みを変えてみること。更に余裕があったら時間発展を 4 次の古典 Runge-Kutta 法に置き換えることを試みよ。

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  Wave equation solver                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

        parameter (n=20,m=50)
c       n: システムサイズ、m: 時間ステップの終り。

        real u(0:n),uu(0:n),uuu(0:n)
        external f,g

              dx=1./real(n)
              dt=1./real(m)

              a=(dt/dx)**2

          do 1 j=0,n
            u(j)=f(real(j)*dx)
 1        continue

           write(6,1000) (dx*real(k),u(k),k=0,n)
           write(6,200)

          do 2 j=1,n-1
            uu(j)=u(j)+dt*g(real(j)*dx)+a*(u(j+1)-2.*u(j)+u(j-1))/2.
 2        continue

          uu(0)=0.
          uu(n)=0.
          uuu(0)=0.
          uuu(n)=0.
```

```
          do 100 i=1,m
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      Main loop to 100                                          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

           do 10 j=1,n-1
             uuu(j)=2.*uu(j)-u(j)+a*(uu(j+1)-2.*uu(j)+uu(j-1))
 10        continue

           do 20 j=0,n
             u(j)=uu(j)
             uu(j)=uuu(j)
 20        continue

       IF(MOD(i,5).EQ.0) THEN
           write(6,1000) (dx*real(k),U(k),k=0,n)
           write(6,200)
       endif

 100      continue

 200      format()
 1000     FORMAT(5x,2f10.4)
         stop
         end

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

           real function f(x)
             f=2.0*x*(1.-x)
           end

           real function g(x)
             g=0.0
           end
```

以上はFortranのプログラム、Cのプログラムは

```
**********************************************************
  シミュレーション概論

  Wave equation solver
```

```
  コンパイルは、
  gcc tawave.c -lm

*********************************************************/
#include <stdio.h>

#define N    20
#define M    50

double f(double);
double g(double);

main()
{
  int i, j, k;
  double a, dx, dt;
  double U[N+1], UU[N+1], UUU[N+1];

  dx = 1.0 / (double)N;
  dt = 1.0 / (double)M;
  a = (dt / dx) * (dt / dx);

  for(i = 0; i <= N; i++){
    U[i] = f((double)i * dx);
  }

  for(k = 0; k <= N; k++){
    printf("\t%10.4f\t%10.4f\n", dx*(double)k, U[k]);
  }
  printf("\n");

  for(i = 1; i <= N - 1; i++){
    UU[i] = U[i] + dt * g((double)j * dx)
      + a * (U[i+1] - 2.0 * U[i] + U[i-1]) / 2.0;
  }

  UU[0] = 0.0;
  UU[N] = 0.0;
  UUU[0] = 0.0;
  UUU[N] = 0.0;

  for(i = 1; i <= M; i++){
    for(j = 1; j <= N-1; j++){
```

```
      UUU[j] = 2.0 * UU[j] - U[j] + a * (UU[j+1] - 2.0 * UU[j] + UU[j-1]);
    }

    for(j = 0; j <= N; j++){
      U[j] = UU[j];
      UU[j] = UUU[j];
    }

    if((i % 5) == 0){
      for(k = 0; k <= N; k++){
printf("\t%10.4f\t%10.4f\n", dx*(double)k, U[k]);
      }
      printf("\n");
    }
  }
}

double f(double x)
{
  return 2.0 * x * (1.0 - x);
}

double g(double x)
{
  return 0.0;
}
```

結果は例えば弦の振動として図１のようになる。

13.3 安定性の条件

ここで差分解の安定性の条件を前章と同様に調べてみよう、差分解として平面波を仮定する。

$$U_j^n = s^n \exp[ikj\Delta x] \tag{9}$$

この式を (1) 式に代入すると ($c = 1$)、次の特性方程式を得る。

$$s^2 - 2s(1 - 2\alpha \sin^2(\frac{k\Delta x}{2})) + 1 = 0 \tag{10}$$

ここで$\beta = \alpha \sin^2(\frac{k\Delta x}{2})$ とおくと、(10) の解は

$$s = 1 - 2\beta \pm \sqrt{4\beta(\beta - 1)} \tag{11}$$

であることがわかる。後の都合上、(13.3) の正号を取るものを s_1, 負号を取るものを s_2としておこう。計算が破綻しないためには $|s_1| \leq 1$ and $|s_2| \leq 1$ を満たす必要がある。仮に$\beta > 1$ とす

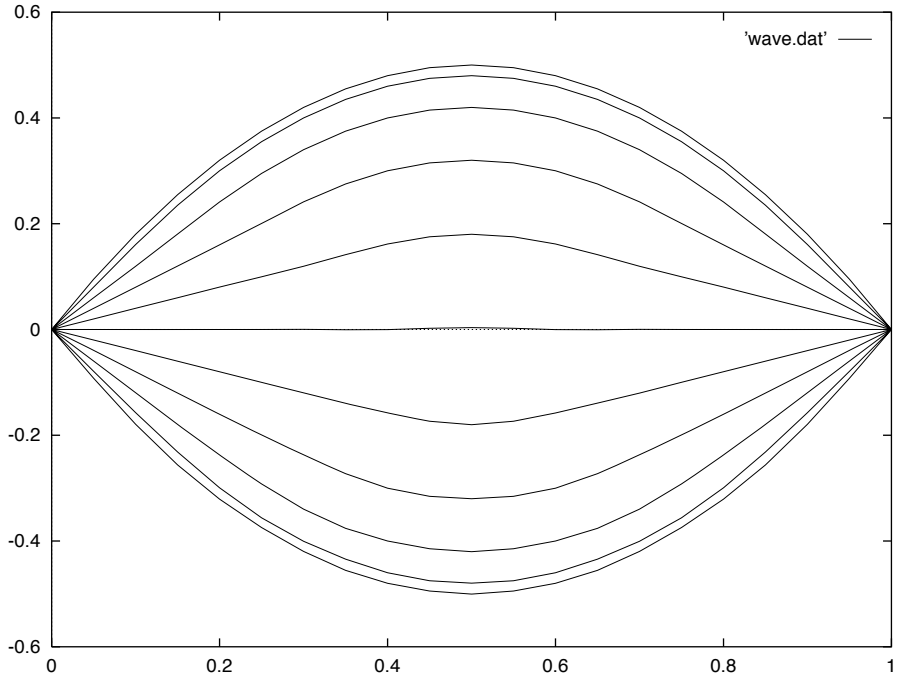

図 1: 固定境界の波動方程式の解。$n = 0, 5, 10, \cdots, 50$。但し$\Delta t = 1/N = 1/50$, $\Delta x = 1/20$.

ると $s_1 \neq s_2$であると同時に $s_1 s_2 = 1$ なので上記の条件は満たさない。一方、$\beta = 1$ であれば $s_1 = s_2 = 1$, $0 < \beta < 1$ であれば$-1 < 1 - 2\beta = Re(s_1) = Re(s_2) < 1$ であって上記の条件を満たす。よって安定性の条件は$\beta \leq 1$ である。書き換えると

$$\alpha \sin^2(\frac{k\Delta x}{2}) \leq 1 \tag{12}$$

であって、任意の kでこの条件が実現するためには$\alpha \leq 1$ 即ち

$$\frac{\Delta t}{\Delta x} \leq 1 \tag{13}$$

が必要となる。

13.4 境界条件

さて今までの例では固定境界条件を扱った。しかし典型的には次の 3 種類の境界条件を扱う必要が生じる場合が多い。従って今までの扱いでは不十分である場合がある。

- 固定境界 (Dirchlet) $u(x,t)$ が境界で与えられる。
- 自由境界 (Neumann) $\partial_x u(x,t)$ が境界で与えられる。
- 周期境界 $u(0,t) = u(L,t)$ など、周期 L で元に戻る。

狭義の自由境界は$\partial_x u = 0$ であろうが、ここではより一般化した Neumann 型の問題を全て自由境界と呼ぶことにしよう。また固定境界と自由境界を組み合わせた境界条件も用いられるし、周期境界は大きな系の一部と見做し、境界の影響が現れにくい場合によく用いられる。実際には 1 次元であれば ring、2 次元であれば torus をシミュレートしていることになる。

境界の影響は重要であるのは高校の物理で学習したと思う。一例を波動方程式にとってみると固定端では入射波に対して位相がπずれた反射波が現れるのに対して、自由端では位相のずれはない。

さて固定境界では既に何回も使っている様に端点の値を固定しておくだけだから計算の際にも支障はない。自由境界の場合は端点とその次の値の差が空間刻み0の極限で微分と一致することを利用する。特に、端点で微分が0である場合には端点と隣の点の値を等しくおくと良い。つまりシステムサイズNの差分系では時間ステップnの場の量に対して$U_1^n = U_2^n$, $U_{N-1}^n = U_N^n$を常に課すのである。

周期境界の場合はもっと簡単である。この場合は端点の更に先に余計に格子点を設けてそれをシステムサイズの剰余系としてみなすのである。すなわちシステムサイズNの差分系では時間ステップnの場の量に対して格子点$0, N+1$を設けて$U_0^n = U_N^n$, $U_{N+1}^n = U_1^n$を常に課すのである

もうプログラムは載せることはしないでも諸君は大丈夫であろう。ここでは初期条件として

$$\phi(x) = 1/\cosh((x-0.5)/0.1)^2$$

として様々な境界条件について時空プロットをしてみた。時間はやはり$t=1$まで、システムサイズは1、刻みは空間が$1/100$, 時間が$1/200$である。データは媒介変数である時間を通して3つのデータの組に格納される。一例として周期境界のFortranのプログラムだけを載せておこう。

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  Wave equation solver                                    C
C  With the periodic boundary condition                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

        parameter (n=100,m=200)
        real u(0:n),uu(0:n),uuu(0:n)
        external f,g

              t=0.0
              dx=1./real(n)
              dt=1./real(m)

              a=(dt/dx)**2

          do 1 j=1,n-1
            u(j)=f(real(j)*dx)
 1        continue
            u(n)=u(1)
            u(0)=u(n-1)

c           write(6,1000) (t,dx*real(k),u(k),k=0,n,4)
           write(6,1000) (t,dx*real(k),u(k),k=0,n)
           write(6,200)
```

```
          do 2 j=1,n-1
            uu(j)=u(j)+dt*g(real(j)*dx)+a*(u(j+1)-2.*u(j)+u(j-1))/2.
 2        continue

           uu(n)=uu(1)
           uu(0)=uu(n-1)

          do 100 i=1,m
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      Main loop to 100                                          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
           t=t+dt
           do 10 j=1,n-1
             uuu(j)=2.*uu(j)-u(j)+a*(uu(j+1)-2.*uu(j)+uu(j-1))
 10        continue

           uuu(n)=uuu(1)
           uuu(1)=uuu(n-1)

           do 20 j=1,n-1
             u(j)=uu(j)
             uu(j)=uuu(j)
 20        continue
             u(n)=u(1)
             u(0)=u(n-1)
             uu(n)=uu(1)
             uu(0)=uu(n-1)

       IF(MOD(i,8).EQ.0) THEN
c           write(6,1000) (t,dx*real(k),U(k),k=0,n,4)
           write(6,1000) (t,dx*real(k),U(k),k=0,n)
           write(6,200)
       endif

 100      continue

 200      format()
 1000     FORMAT(5x,3f10.4)
         stop
         end

ccccccccccccccccccccccccccccccccccccccccccccc
```

```
real function f(x)
  f=1.0/cosh((x-0.5)/0.1)**2
end

real function g(x)
  g=1.0
end
```

他の境界条件や C 言語のプログラムは省略する。

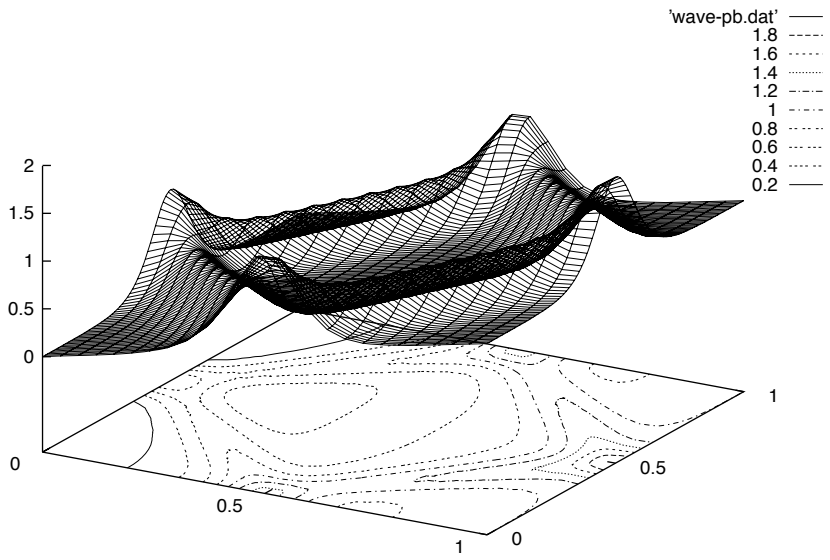

図 2: 周期境界の波動方程式の解。但し$\Delta t = 1/N = 1/200$, $\Delta x = 1/100$. 表示は $8\Delta t$ 毎に時空の２次元データとして格納している。

ここで表示を工夫してみよう。いわゆる３次元プロットと等高線プロットを同時に行なう。gnuplot に入って

gnuplot> set parametric

とタイプしてみよう。ここではデータが媒介変数を通して格納されているので parametric mode に入る必要がある。ここで

gnuplot> sp 'wave-pb.dat' w l

とタイプすると網目状のデータが見える筈である。この場合、時間が横軸、空間が縦軸となって、その時空点でのデータがプロットしてある。拡張子 w l は with line の省略形である。

gnuplot> set hid

gnuplot> replot

とすると見やすくなる。se hid で隠線処理、すなわち隠れた線の表示をしない様にしたのである。replot は同じ絵を再描画するときに用いる。角度などは好き勝手にいじれば良いが on-line manual を参照のこととして省略する。ここでは次に等高線を描くことにトライする。

gnuplot> set contour

gnuplot> replot

とすると下に線が出てくる。contour が等高線を呼び出す命令である。等高線の値は Caption として出ている。いかにも間隔がまずいので

gnuplot> set cntrparam levels incremental 0,.2,2.

gnuplot> replot

とタイプしてみよう。cntrparam はパラメータのコントロールに使う。様々な機能があるので manual で各自確認しておくこと。ここで 1 番初めの数字は基点（最小点）、次が等高線の刻み、最後の数字は最大値である。

次は既に行なった固定境界条件である。随分結果が違ってきているのがわかる。自由境界では

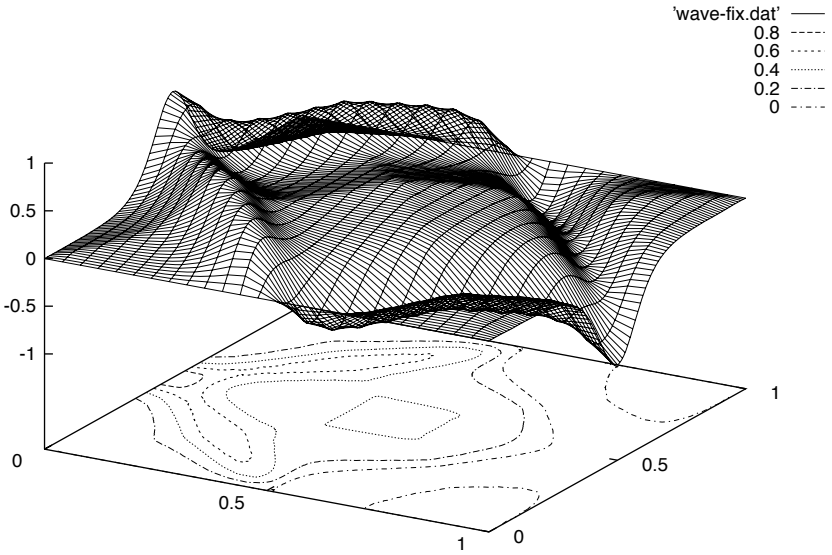

図 3: 固定境界の波動方程式の解。但し$\Delta t = 1/N = 1/200$, $\Delta x = 1/100$. 表示は $8\Delta t$ 毎に時空の２次元データとして格納している。

どうか。

あまり周期境界の場合と差がない。これは波動方程式でミラーサイトから走ってくる波が自由境界での反射波と同じ役割をしていることから理解できよう。最後に片側を自由境界、もう片側を固定境界にしてみるとどうなるか。ここでは $x = 0$ を固定、$x = 1$ を自由境界とする。

以上の様に一般に偏微分方程式の解は境界の影響を強く受ける。従ってその取り扱いには充分な注意が必要である。

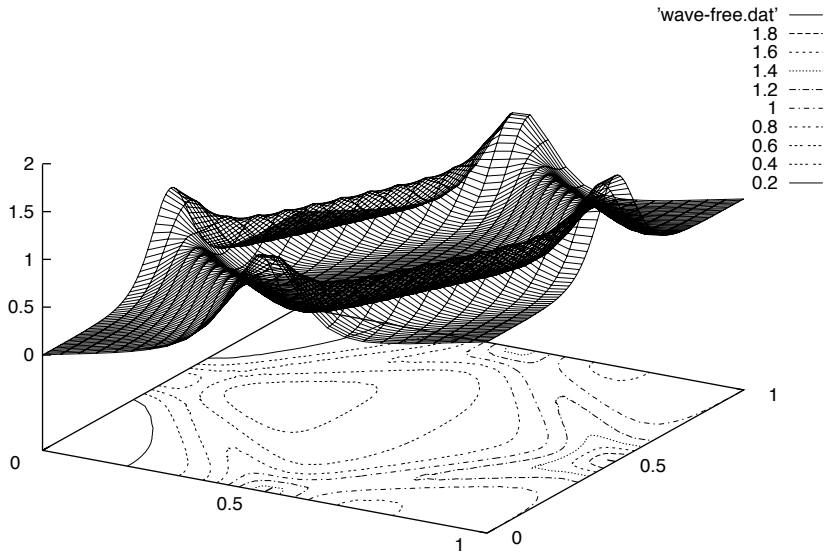

図 4: 自由境界の波動方程式の解。但し$\Delta t = 1/N = 1/200$, $\Delta x = 1/100$. 表示は $8\Delta t$ 毎に時空の２次元データとして格納している。

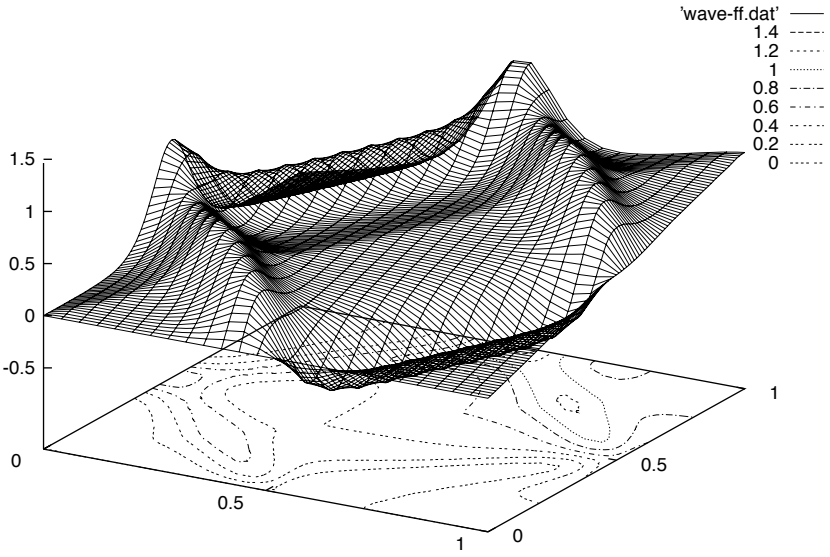

図 5: 固定境界と自由境界の混合条件下での波動方程式の解。但し$\Delta t = 1/N = 1/200$, $\Delta x = 1/100$. 表示は $8\Delta t$ 毎に時空の２次元データとして格納している。