S-W近似によって様々な領域の熱方程式を解くためのアルゴリズム

明治大学　理工学部　数学科
金子　裕司
指導教諭　桂田祐史

2007年2月20日

目 次

第I部

S-W近似によって円盤領域の熱方程式を解くためのアルゴリズム

第1章　S-W近似とは

1.1　S-W近似に関するイントロ

S-W近似の正式な名前は「Shortley-Weller(ショートリィ・ウェラー)近似」といいます。以下、S-W近似と呼んでいきます。一般的な領域に対しては、適当な写像で長方形領域に変換してから差分法を適用する（円盤領域では極座標によって変換できる）などの工夫がなされています。しかし、このS-W近似では長方形領域に変換をせずにそのままの形で差分法を行っていきます。考える領域Ωの形状にかかわらずに、十分細かい差分格子をつくりS-W近似によって解くことが出来ます。また、今後考えていく熱方程式の初期値境界地問題は以下のようであるとします。

$$\begin{cases} u_t = \Delta u & ((x,y) \in \Omega, 0 < t) \\ u = f(x,y) & ((x,y) \in \Omega, t = 0) \\ u = 0 & ((x,y) \in \delta\Omega) \end{cases}$$

1.2　S-W近似とは

Ωを$\mathbf{R^2}$の有界領域とし、刻み幅$h_x = h_y = h$の等間隔差分格子点をつくる。境界付近の上下左右の格子点がΩの閉包の外に出てしまうことがあります。その場合に、等間隔格子点であることをあきらめて、境界上に格子点を新しく定めます。そうして出来た不等間隔格子点に対しても差分法によって近似してく方法をS-W近似といいます。

領域をx座標、y座標ともにN等分します。ある格子点を取ったとき、その上下左右の点とその距離の表し方は下の図2のように表していきます。

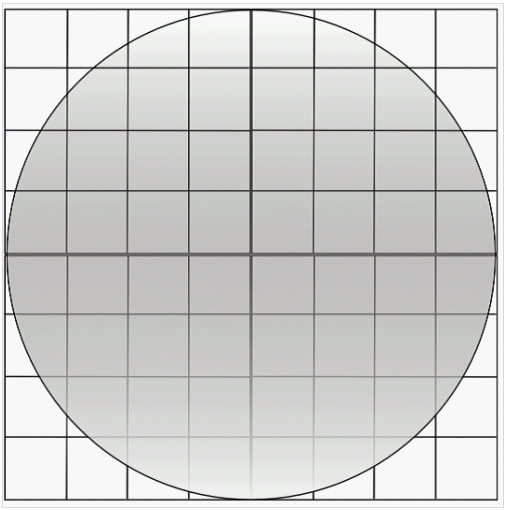


図1、正方形領域内の円盤

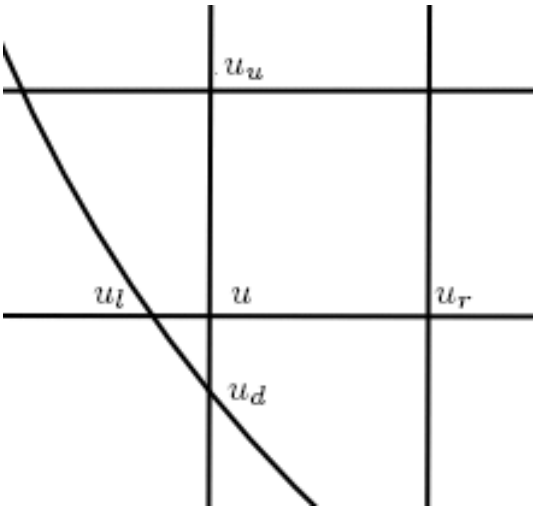


図2、上下左右の点の値の呼び方

つまり、ある格子点上の点 u_{ij} を取ってきたときに左の点を u_l とし、u_{ij} と u_l との距離を h_l とします。同様に上下と右にも同じように名前を付けます。通常は $h_l = h_r = h_u = h_d = h$ となっていますが、不等間隔格子点がある場合にはそうなりません。S-W 近似の 2 次元の場合には式は下のように与えられます。

$$\Delta u \simeq \frac{2}{h_l + h_r}\left(\frac{u_r - u}{h_r} - \frac{u - u_l}{h_l}\right) + \frac{2}{h_u + h_d}\left(\frac{u_u - u}{h_u} - \frac{u - u_d}{h_d}\right).$$

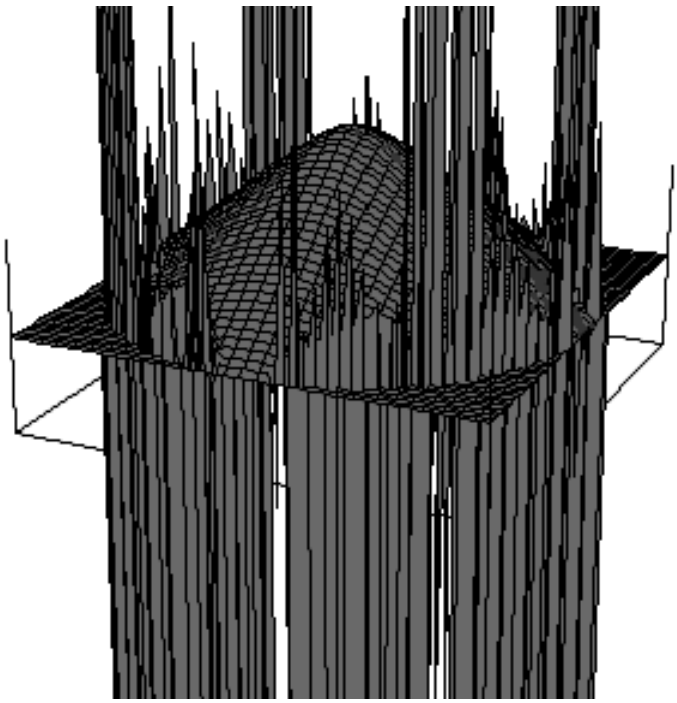
$h_l = h_r = h_u = h_d = h$ であれば通常の差分方程式となります。

$$\Delta u \simeq \frac{1}{h^2}(u_u + u_d + u_r + u_l - 4u).$$

1.3 τ の取り方による崩壊の様子

この次で安定性条件を導くことになりますが、その前に τ の取り方によるグラフの変化を見ていきましょう。

$N = 30$ の場合に $\tau \leq \frac{h^2}{4} = 6.9444e-05$（通常の差分近似の安定条件）なので $\tau = 6.9444e-05$ で取ってしまうと



$t = 0.001$ の様子ですが、上のように不安定になります。境界部分の値から激しく振動してしまって、そこからすぐに全体に広がっていることがわかります。

τ を次で求める安定性条件下におけばこういった崩壊が起きません。この場合ですと $\tau \leq \frac{h^2}{4} = 6.9444e-05$ よりも約 20 倍ほど強く抑えることになります。

1.4 安定条件

τ は時間の刻み幅で、S-W 近似を必要としない正方形領域では

$$\tau \leq \frac{h^2}{4}.$$

のように取れば安定になります。しかし、円盤領域では不等間隔格子点が発生し h よりも小さな h_r, h_l, h_u, h_d が存在します。なので、この $\tau \leq \frac{h^2}{4}$ よりも強く抑えないと安定にすることが出来ません。以下、どのように τ を抑えればよいか考えていきます。

円盤の場合、格子点上の点を一つ取りその上下左右が不等間隔になりえる点は高々2 点で、上下左右の全てが不等間隔であることはありえません。このことから、上下と左右のうち一つずつが不等間隔であったとします。そのときの間隔は h よりも小さいので、それぞれ $\varepsilon_1 h, \varepsilon_2 h (0 < \varepsilon_1, \varepsilon_2 < 1)$ とします。この $\varepsilon_1 h, \varepsilon_2 h$ を

$$\Delta u \simeq \frac{2}{h_l + h_r}\left(\frac{u_r - u}{h_r} - \frac{u - u_l}{h_l}\right) + \frac{2}{h_u + h_d}\left(\frac{u_u - u}{h_u} - \frac{u - u_d}{h_d}\right).$$

に代入していきます。便宜上不等間隔になる点を上と左にし $h_l = \varepsilon_1 h$、$h_u = \varepsilon_2 h$ とします。

$$\Delta u \simeq \frac{2}{\varepsilon_1 h + h}\left(\frac{u_r - u}{h} - \frac{u - u_l}{\varepsilon_1 h}\right) + \frac{2}{\varepsilon_2 h + h}\left(\frac{u_u - u}{\varepsilon_2 h} - \frac{u - u_d}{h}\right).$$

これを変形して、

$$\Delta u \simeq \frac{2}{h^2}\left(\frac{\varepsilon_1 u_r + u_l}{\varepsilon_1(\varepsilon_1 + 1)} + \frac{\varepsilon_2 u_d + u_u}{\varepsilon_2(\varepsilon_2 + 1)}\right) - \frac{2(\varepsilon_1 + \varepsilon_2)}{\varepsilon_1 \varepsilon_2 h^2} u.$$

$u_t = \Delta u, u_t = \frac{u^{n+1} - u^n}{\tau}$ を用いて、u^{n+1} について変形する。
ここで、今までの u は u^n であるので、それを省略せずに書くことにする。

$$u^{n+1} \simeq \frac{2\tau}{h^2}\left(\frac{\varepsilon_1 u_r^n + u_l^n}{\varepsilon_1(\varepsilon_1 + 1)} + \frac{\varepsilon_2 u_d^n + u_u^n}{\varepsilon_2(\varepsilon_2 + 1)}\right) + \left(1 - \frac{2\tau(\varepsilon_1 + \varepsilon_2)}{\varepsilon_1 \varepsilon_2 h^2}\right) u^n.$$

ここで過去の卒研によると、差分方程式の右辺に現れる係数は正であることが安定性のために必要です。この場合も数値実験の結果によると、u^n の係数が正にならなければならないので、τ について整理すると

$$\tau \leq \frac{\varepsilon_1 \varepsilon_2 h^2}{2(\varepsilon_1 + \varepsilon_2)}.$$

で表すことが出来ます。これで、τ の条件を得ることが出来ました。
高々2 点が境界に乗ることを保障せずに、4 点全てが境界に乗る可能性を考慮すると以下のようになります。

$$\tau \leq \frac{\varepsilon_1 \varepsilon_2 \varepsilon_3 \varepsilon_4 h^2}{2(\varepsilon_1 \varepsilon_3 + \varepsilon_2 \varepsilon_4)}.$$

第2章　アルゴリズム

2.1　境界上の点の探索方法

上下左右の点を考える必要がありますが、ここでは左の場合を例に書いていきます。

円盤領域では、中心からの距離が半径未満ならばその点は円盤内部に存在していることがわかり、逆に円盤外部ということは中心からの距離が半径よりも大きいことになります。つまり、境界を見つけるときには円盤内部の点に対してその一つだけ隣の点が外部にあるのなら境界が間に存在しているとわかります。

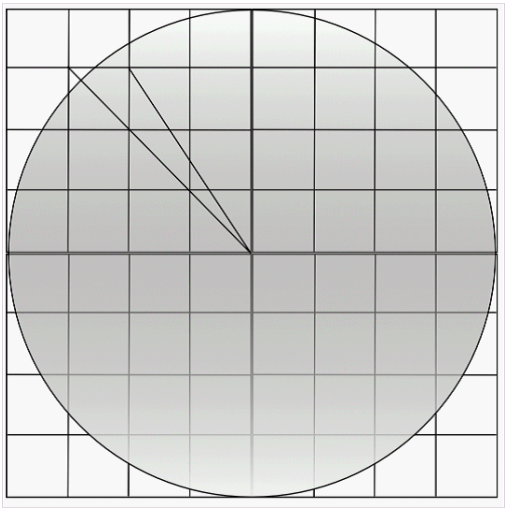


図 2.1 境界判定

このことを利用して、円盤内部の点全てに対し、その一つ左の点が外に出ていないかどうかを判断して境界が間にあるかどうかを調べます。具体的には u_{ij} と原点との距離が半径よりも小さい点（円盤内部）の全ての点に対して、u_l(隣の格子点) と原点との距離が半径よりも大きい点を探すことになります。境界が間にあると判断した際には、境界の座標を計算しなければなりません。

2.2　境界の座標の計算

境界が点と点の間に存在していることがわかっただけではまだ計算に使えないので、そこから具体的に座標の値を計算していくことになります。通常格子点の上ではその x、y 座標は共にhの倍数になっていますが、境界上にある不等間隔格子点では x、y 座標のどちらか一方のみが h の倍数となります。このことから、三平方の定理を用いて座標を求めていきます。また、u_{ij} の座標を (x_i,y_j) とします。

上の例は左側が境界上に乗っている場合ですが、半径 r と h に依存している y 座標から不等間隔になる x を

$$x = \sqrt{r^2 - y_j^2}.$$

で求めることが出来ます。これによって u_{ij} に対する u_l の座標が（$x = \sqrt{r^2 - y_j^2}$ 、y_j) で表されることがわかります。また、u_{ij} と u_l との距離である h_l は $h_l = x - x_i$ で求めることが出来ます。同様のことを、上下右にも行います。

2.3 丸め誤差対策

実験を繰り返すうちに丸め誤差のせいで実際には境界上の点でも円の内部にあると判断してしまうことがありましたので、丸め誤差対策をプログラム内部で行っています。

具体的には領域を 10 個に刻んだ半径 5 の円盤の場合に、三角比で有名な $(3/N)^2 + (4/N)^2 = (5/N)^2$ のような格子点が発生して、円の境界が格子点に乗ります。この場合に丸め誤差が発生して PC 内部では $(3/N)^2 + (4/N)^2 = (5/N)^2 - \alpha$ になっています。

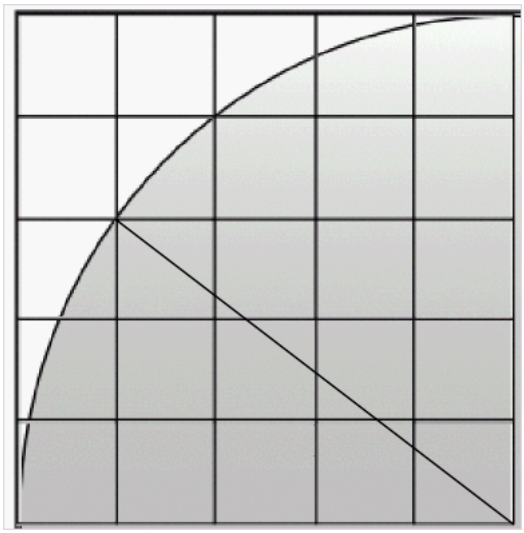


図 2.3 境界が格子点に乗っている場合

このとき、何の対策もしなければ円の内部にあると勘違いして極端に小さい ε を返します。念のために書いておけば double 型では 10^{-15} 付近の誤差となります。このために、境界が格子点に乗っている場合に、丸め誤差の影響を考えて内部ではないと判定しなければなりません。

このプログラムでは丸め誤差対策として半径を本来の r ではなく $r - 10^{-14}$ としてわずかに小さくすることで、本来は境界の点が誤って内部にあると判定することを防ぐことが出来ます。

この方法では、境界が極めて近い場合も適用されてしまいますが、実験では $N = 500$（領域 1000 等分）までで ε は通常 10^{-3} 以上なので 10^{-14} と大きく差があり問題ないと判断しました。

N を大きくすると計算量は大きくなり 100 でも計算にとても時間がかかるので、$N > 500$ はとても計算できません。

2.4　ε の値

丸め誤差で頻繁に出てくる ε について具体的にどうなっていくのかを調べていきます。ε の中でも τ の安定条件算出に必要な最小 ε を話題にしています。ε は単調現象というわけではありませんが、N を大きくしていくと全体的に小さくなっていることがわかります。ランダムと言ってもどの程度ランダムなのか参考までに N=500 までのデータを (x, y) 座標にプロットしたグラフを掲載します。縦軸が ε、横軸が N をあらわしています。

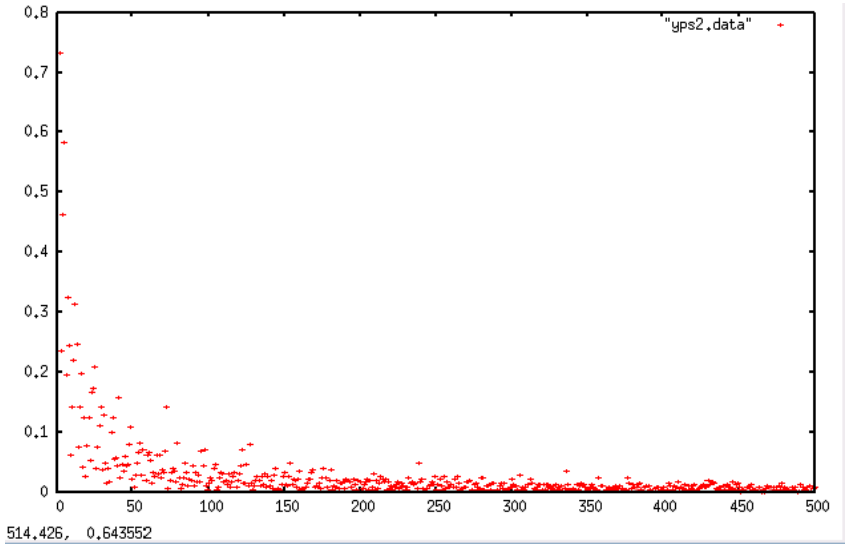


図 2.4.1 ε の分布

次に、このグラフを対数グラフにしたものを掲載します。

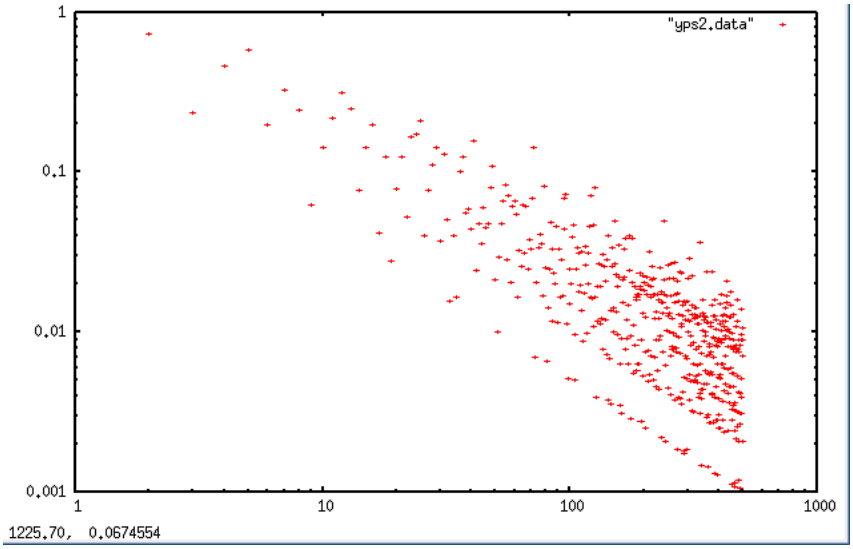


図 2.4.1 ε の分布 (対数)

このグラフから $\varepsilon \geq \frac{c}{N}$(ただし c は定数) と書けると予想できます。

2.5　円盤の特性

上下左右の全てが不等間隔になるということはなく、そのうち高々2 個の不等間隔格子点が存在することがわかります。左半円では円盤内の格子点を取ったとき右隣の格子点は境界の外に出るということはありません。このことは同様に右半円では、左の点が等間隔であることを示し、さらに上半円では下の点が、下半円では上の点が等間隔で円盤内に入っていることがわかります。このことを利用し、プログラム内では 4 つのパートに分けて計算しています。4 つのパートとは、((左下)(左上))((右下)(右上))です。つまり、左上では円盤内の点の左と上に境界がないかを調べて境界があれば不等間隔格子点を作り、右と下はそのまま格子点を与えればいいわけです。

2.6　初期値の設定

この前までの章で熱を与えたときの時間経過に対する熱の変異を調べていきましたが、この章では熱を与える一番初めのことを書きます。このプログラムで与えている初期値は最も一般的な山型を与えています。具体的な式は $f(x,y) = 1 - \sqrt{x^2+y^2}$ となります。

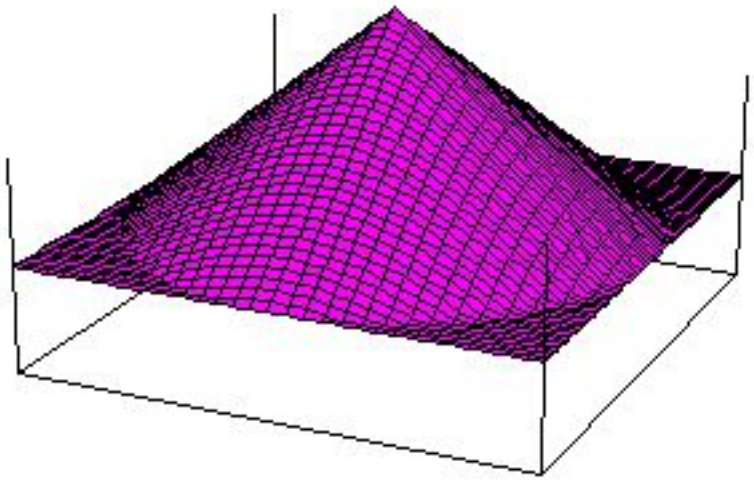
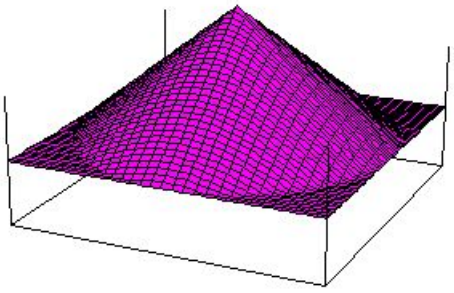


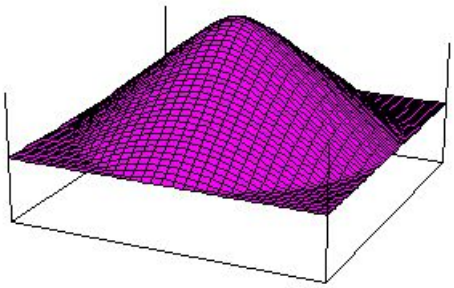
図　2.6 円錐型の初期値

第3章　実験結果

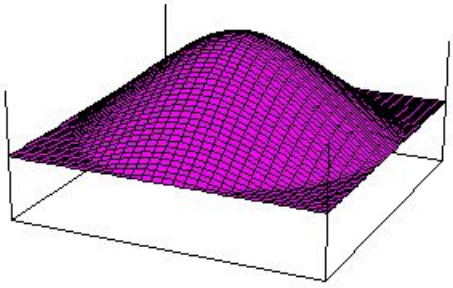
N=30(領域全体 60 等分)



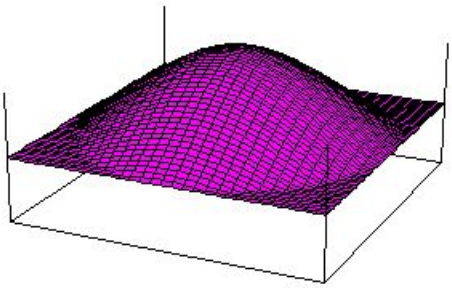
t=0



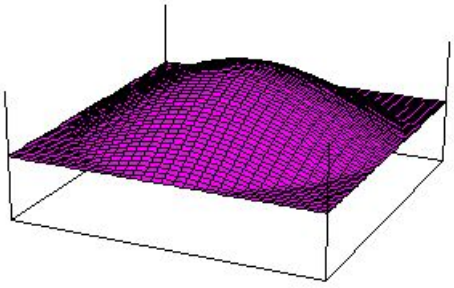
t=0.001



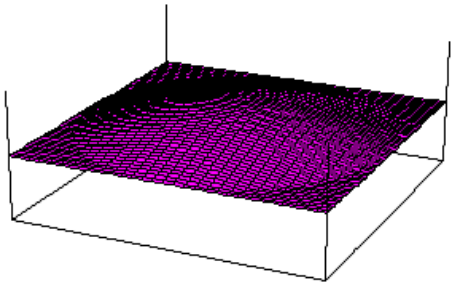
t=0.005



t=0.01



t=0.02



t=0.05

第4章　プログラムリスト

4.1　S-W 近似プログラム

S-W 近似による円盤領域の熱方程式を解くプログラム本文です。

```
/*
 * heat2d-e-sw.c -- solve the heat equation in a two dimensional square region
 * by explicit finite difference method. (version 6.0)
 *
 *      To compile this program  ->  ccmg heat2d-e-sw.c
 */

#include <stdio.h>
#include <math.h>

/* to use matrix, new_matrix() */
#include <matrix.h>

/* to use GLSC */
#define G_DOUBLE
#include <glsc.h>


int main()
{
    int             N, i, j, n, skip, nMax;
    double          h, lambda, tau, Tmax, t, c, dt ,yps1,yps2,ypsr,minyps,maxtau;
    double          x,r,hl,hr,hu,hd;
    double          xi,xil,xir,yj,yjd,yju;
    double          ul,ur,uu,ud;
    matrix          u, newu;
    double          f(double, double, double);
    double          a(double, double);

    printf("N: "); scanf("%d",&N);
    N=2*N;
```

```
h = 1.0 / N;
r=0.5;
yps1=1.0;
yps2=1.0;
maxtau=1.0;
minyps=1.0;
ypsr=1.0e-14;

 /****************最小εと最大τを求める ****************/
  for (i = 1;i<N/2; i++){
    xi =r-i*h;
    xil=r-(i-1)*h;
    for (j = 1;j<N/2; j++){
      yj=r-j*h;
      if(sqrt(xi*xi+yj*yj)-r<-ypsr && sqrt(xil*xil+yj*yj)>r){
        yps1=(sqrt(r*r-yj*yj)-xi)/h;

        if(yps1<minyps)
          minyps=yps1;

        hd=sqrt(r*r-xi*xi)-yj;

        if(hd>h){
          hd=h;
        }
        yps2=hd/h;

        if(maxtau>yps1*yps2*h*h/(2.0*(yps1+yps2)))
          maxtau=yps1*yps2*h*h/(2.0*(yps1+yps2));
      }
    }
  }
  printf("εの最小値=%g\n",minyps);

  if(minyps<1.0e-06){
    printf("*******εが小さすぎます********\n");
  }

if ((u = new_matrix(N + 1, N + 1)) == NULL) {
    fprintf(stderr, "配列 u を確保できませんでした。");
    exit(1);
}
```

```
if ((newu = new_matrix(N + 1, N + 1)) == NULL) {
    fprintf(stderr, "配列 newu を確保できませんでした。");
    exit(1);
}
printf("Tmax: "); scanf("%lf", &Tmax);

printf("τ (≦%g ): ", maxtau);
scanf("%lf", &tau);


lambda = 2.0*tau/(h*h);
printf("λ=%g になりました。\n", lambda);

printf("Δ t: ");
scanf("%lf",  &dt);
skip = rint(dt / tau);
if (skip == 0) {
  printf("Δ t が小さすぎるので、Δ t=τ とします。\n");
  skip = 1;
}
dt = skip * tau;

g_init("Meta", 250.0, 160.0);
g_device(G_BOTH);
g_def_scale(0, 0.0, 2.0, 0.0, 2.0, 30.0, 70.0, 100.0, 72.0);
g_def_line(0, G_BLACK, 0, G_LINE_SOLID);
g_sel_scale(0);

for (i = 0; i <= N; i++)
    for (j = 0; j <= N; j++)
        u[i][j] = f(i*h, j*h,r);

g_cls();
g_hidden2(1.0, 1.0, 0.4, -1.0, 1.0, 5.0, 25.0, 20.0, 20.0, 20.0,
    150.0, 100.0, u, N + 1, N + 1,
    1, G_SIDE_NONE, 2, 1);

nMax = rint(Tmax / tau);

for (n = 1; n <= nMax; n++) {
  /**********************左半円**********************/
  for (i = 1; i < N/2; i++){
    xi =r-i*h;                          /* u[i][j] の x 座標 */
```

```
xil=r-(i-1)*h;                          /* u[i][j] の左の点の x 座標*/
/**********************円盤領域左下**********************/
    for (j = 1; j < N/2; j++){
      yj=r-j*h;                         /* u[i][j] の y 座標 */
      yjd=r-(j-1)*h;                    /* u[i][j] の下の y 座標 */

      if(sqrt(xi*xi+yj*yj) < r-ypsr){
        if(sqrt(xil*xil+yj*yj) > r-ypsr){
          x=sqrt(r*r-yj*yj);
          hl=x-xi;
          ul=a(r-x,j*h);
        }
        else{
          hl=h;
          ul=u[i-1][j];
        }
        hr=h;
        ur=u[i+1][j];
    }
    if(sqrt(xi*xi+yj*yj) < r-ypsr){
      if(sqrt(xi*xi+yjd*yjd) > r-ypsr){
        x=sqrt(r*r-xi*xi);
        hd=x-yj;
        ud=a(i*h,r-x);
      }
      else{
        hd=h;
        ud=u[i][j-1];
      }
      hu=h;
      uu=u[i][j+1];

      newu[i][j]=u[i][j]+tau*(2.0*((ur-u[i][j])/hr-(u[i][j]-ul)/hl)/(hr+hl)+
                              2.0*((uu-u[i][j])/hu-(u[i][j]-ud)/hd)/(hu+hd));
    }
    else{
      newu[i][j] = 0.0;
    }
    }

    /**********************円盤領域左上**********************/
  for (j = N/2; j < N; j++){
    yj=j*h-r;
```

```
      yju=(j+1)*h-r;
      if(sqrt(xi*xi+yj*yj) < r-ypsr){
        if(sqrt(xil*xil+yj*yj) > r-ypsr){
          x=sqrt(r*r-yj*yj);
          hl=x-xi;
          ul=a(r-x,j*h);
        }
        else{
          hl=h;
          ul=u[i-1][j];
        }
        hr=h;
        ur=u[i+1][j];
      }
      if(sqrt(xi*xi+yj*yj) < r-ypsr){
        if(sqrt(xi*xi+yju*yju) > r-ypsr){
          x=sqrt(r*r-xi*xi);
          hu=x-yj;
          uu=a(i*h,r+x);
        }
        else{
          hu=h;
          uu=u[i][j+1];
        }
        hd=h;
        ud=u[i][j-1];

        newu[i][j]=u[i][j]+tau*(2.0*((ur-u[i][j])/hr-(u[i][j]-ul)/hl)/(hr+hl)+
                                2.0*((uu-u[i][j])/hu-(u[i][j]-ud)/hd)/(hu+hd));
      }
      else  newu[i][j]=0.0;
    }
}
/**********************右半円************************/
for (i = N/2; i < N; i++){
  xi=i*h-r;
  xir=(i+1)*h-r;
  /**********************円盤領域右下**************************/
    for (j = 1; j < N/2; j++){
      yj=r-j*h;
      yjd=r-(j-1)*h;
      if(sqrt(xi*xi+yj*yj) < r-ypsr){
        if(sqrt(xir*xir+yj*yj) > r-ypsr){
```

```
      x=sqrt(r*r-yj*yj);
      hr=x-xi;
      ur=a(r+x,j*h);
    }
    else{
      hr=h;
      ur=u[i+1][j];
    }
    hl=h;
    ul=u[i-1][j];
  }
  if(sqrt( xi*xi + yj*yj) < r-ypsr){
    if(sqrt( xi*xi + yjd * yjd ) > r-ypsr){
      x=sqrt(r*r-xi*xi);
      hd=x-yj;
      ud=a(i*h,r-x);
    }
    else{
      hd=h;
      ud=u[i][j-1];
    }
    hu=h;
    uu=u[i][j+1];

    newu[i][j]=u[i][j]+tau*(2.0*((ur-u[i][j])/hr-(u[i][j]-ul)/hl)/(hr+hl)+
                            2.0*((uu-u[i][j])/hu-(u[i][j]-ud)/hd)/(hu+hd));
  }
  else  newu[i][j] = 0.0;
}
/**********************円盤領域右上**************************/
for (j = N/2; j < N; j++){
  yj=j*h-r;
  yju=(j+1)*h-r;
  if(sqrt(xi*xi+yj*yj) < r-ypsr){
    if(sqrt(xir*xir+yj*yj) > r-ypsr){
      x=sqrt(r*r-yj*yj);

      hr=x-xi;
      ur=a(r+x,j*h);
    }
    else{
      hr=h;
      ur=u[i+1][j];
```

```
            }
            hl=h;
            ul=u[i-1][j];
          }
          if(sqrt(xi*xi+yj*yj) < r-ypsr){
            if(sqrt(xi*xi+yju*yju) > r-ypsr){
              x=sqrt(r*r-xi*xi);
              hu=x-yj;
              uu=a(i*h,x+r);
            }
            else{
              hu=h;
              uu=u[i][j+1];
            }
            hd=h;
            ud=u[i][j-1];

            newu[i][j]=u[i][j]+tau*(2.0*((ur-u[i][j])/hr-(u[i][j]-ul)/hl)/(hr+hl)+
                                    2.0*((uu-u[i][j])/hu-(u[i][j]-ud)/hd)/(hu+hd));
          }
          else  newu[i][j]=0.0;
        }
      }
        for (i = 0; i <= N; i++)
            for (j = 0; j <= N; j++)
                u[i][j] = newu[i][j];
        if (n % skip == 0){
            g_cls();
            g_hidden2(1.0, 1.0, 0.4, -1.0, 1.0, 5.0, 25.0, 20.0, 20.0, 20.0,
                150.0, 100.0, u, N + 1, N + 1,
                1, G_SIDE_NONE, 2, 1);
        }
        t = tau * n;
    }
    /* マウスでクリックされるのを待つ */
    g_sleep(-1.0);
    /* ウィンドウを消す */
    g_term();
    return 0;
}


double f(double x, double y,double r){
```

```
  if(sqrt((x-r)*(x-r)+(y-r)*(y-r)) < r+1.0e-14 )

    return 2.0-4.0*sqrt((x-r)*(x-r)+(y-r)*(y-r));

  else  return 0;
}


double a(double x, double y){
  return 0;
}
```

4.2 最小 ε 表作成プログラム

指定した N までの ε を羅列します。最後にその中でも最小の ε を出力します。

```
#include <stdio.h>
#include <math.h>

main(){

    int             N, i, j, k;
    double          h,t,c;
    double          yps,minyps,ypsr,minminyps;
    double          x,r,hl;
    double          xi,xil,yj;

    /* printf("N: "); scanf("%d",&N);*/

    N=500;

    r = 0.5;
    yps = 1.0;
    ypsr=1.0e-14;
    minminyps=1.0;

    for(k = 2 ;k<=N; k++){
      k = k*2;
      h = 1.0/k;
      minyps = 1.0;
```

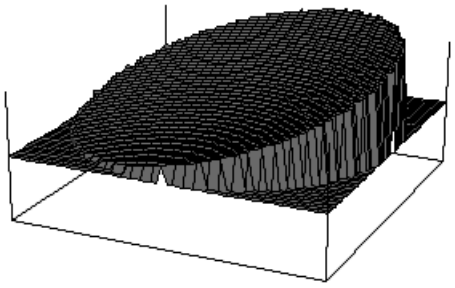
```
      for (i = 1 ; i < k/2 ; i++){
        xi = r-i*h;
        xil = r-(i-1)*h;

        for (j = 1 ; j<k/2 ; j++){
          yj = r-j*h;

          if(sqrt(xi*xi+yj*yj)-r < -ypsr && sqrt(xil*xil+yj*yj)>r){
            yps=(sqrt(r*r-yj*yj)-xi)/h;

            if( yps < minyps ){
              minyps = yps;
              if(minyps < minminyps ){
                minminyps = minyps;
              }
            }
          }
        }
      }
      if((k/2)%5==0){
        printf("N=%3d    εの最小値=%g    ", k/2 , minyps);
      }
      k=k/2;
    }
    printf ("\n 最小最小ε＝%g     \n",minminyps);
}
```

第5章　様々な初期値とその実験結果

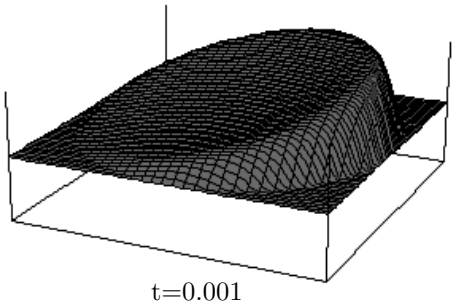
今まで考えてきた初期値は回転対象であり単純であったため、もっと工夫した初期値にしていきます。

5.1　$x^2 + y^2$

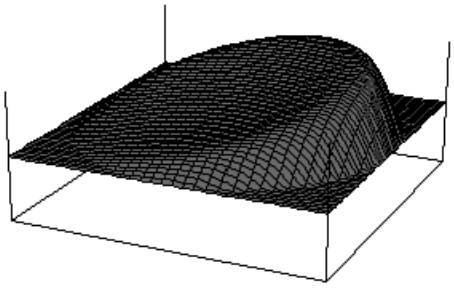
N=30(領域全体 60 等分)



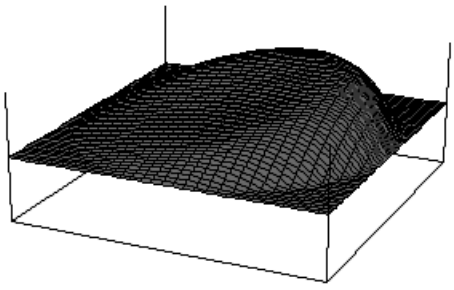
t=0



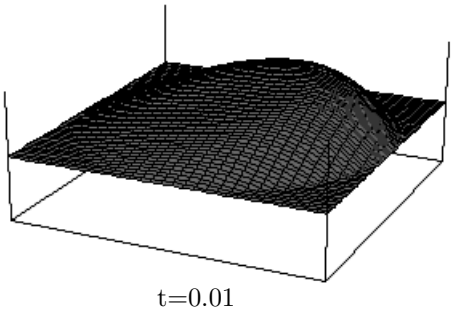
t=0.001



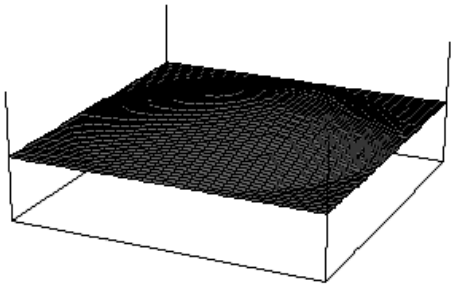
t=0.002



t=0.005



t=0.01



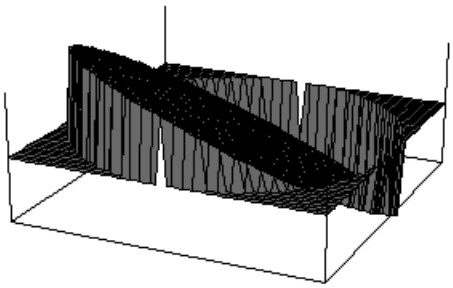
t=0.03

5.2 $1 - x - y$

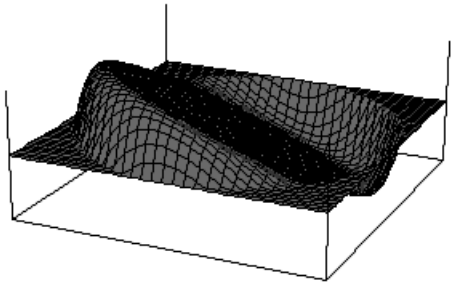
$x^2 + y^2$ や最初の円錐形では正の値のみを取っているので、ここでは正負の両方を与える初期値を選びます。初期値の変更は下のように変更しています。

```
return  2-2*(x+y);
```

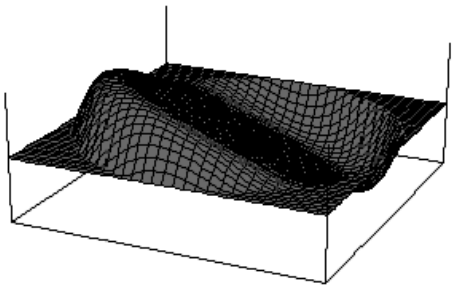
N=30(領域全体 60 等分)



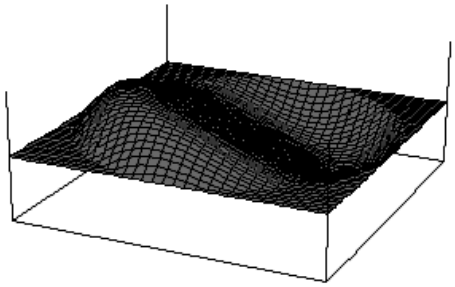
t=0



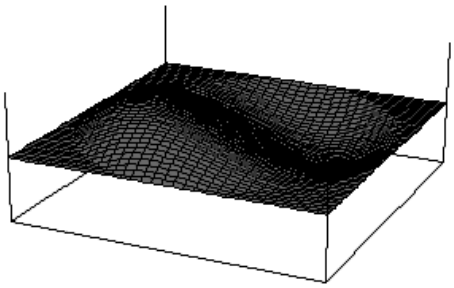
t=0.001



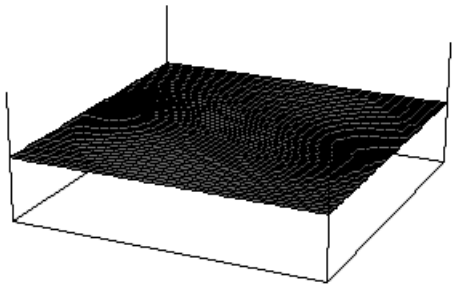
t=0.002



t=0.005



t=0.01



t=0.03

第II部

S-W近似によってCassiniの橙形領域の熱方程式を解くためのアルゴリズム

第6章　Cassiniの橙形とは

この2部では円盤領域よりも複雑なCassiniの橙形の1パターンであるひょうたん型領域におけるS-W近似を見ていきます。
そのためにもまずはCassiniの橙形なるひょうたん型領域を知らなければなりません。

Cassiniの橙形とはx座標に点$\pm a(a$は正の定数)があり、その2点からの距離の積がbとなる点を集めた曲線のことをいいます。

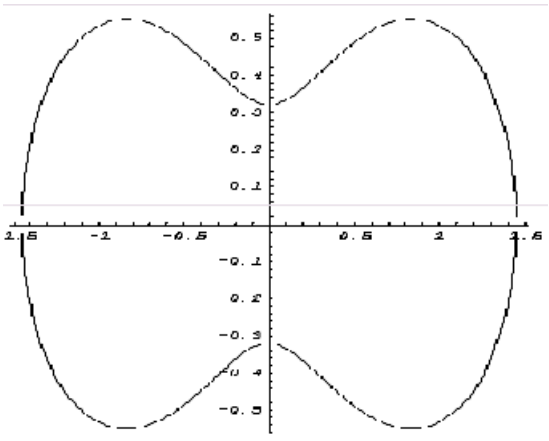
Cassiniの橙形はa, bの大小関係で以下のように姿を変えます。

$$\begin{cases} \cdot a > b & \Rightarrow \text{二つの連結成分} \\ \cdot a = b & \Rightarrow \infty\text{型} \\ \cdot \sqrt{2}a > b > a & \Rightarrow \text{凹みのある閉曲線 (ひょうたん型)} \\ \cdot b \geq \sqrt{2}a & \Rightarrow \text{凸閉曲線} \end{cases}$$

具体的な式で表すと

$$(x^2 + y^2 + a^2)^2 = 4a^2x^2 + b^4.$$

となります。ここでは、$\sqrt{2}a > b > a$のひょうたん型のみ見ていくことにします。ひょうたん型の場合下のグラフのようになります。(a=1,b=1.05)



ひょうたん型

第7章　円盤からのアルゴリズムの変更点

7.1　境界判定

境界判定はいままでと同様に領域内部が不等式で表せるので円盤内部が

$$\sqrt{x^2+y^2} < r.$$

だった所を Cassini の橙形の内部判定では

$$\sqrt{(x^2+y^2+a^2)^2-4a^2x^2} < b^2.$$

としています。ここでルートに入れなくても特に違いはありません。またこの条件が非常に長いので外部関数定義しておきます。この条件の下に一部と同様に隣の点が外部かどうかを調べて間に境界があるかどうかを判定します。

7.2　座標計算

Cassini の橙形 (ひょうたん領域の場合) は 4 価関数なので注意が必要です。

$$\sqrt{(x^2+y^2+a^2)^2-4a^2x^2} < b^2.$$

Cassini の橙形の式を x について解きます。

$$x = \pm\sqrt{a^2-y^2 \pm \sqrt{b^4-4a^2y^2}}.$$

となり、また同様に y についても解くと

$$y = \pm\sqrt{-a^2-x^2 \pm \sqrt{b^4+4a^2x^2}}.$$

しかし、$-a^2-x^2 \pm \sqrt{b^4+4a^2x^2} \geq 0$ とならなければならないので

$$y = \pm\sqrt{-a^2-x^2 + \sqrt{b^4+4a^2x^2}}.$$

座標計算はこれらの式を利用して求めることが出来ます。

7.3　安定条件

円盤同様 τ の最大値と ε の最小値を求める必要があります。一部で求めた安定条件は

$$\tau \leq \frac{\varepsilon_1\varepsilon_2\varepsilon_3\varepsilon_4 h^2}{2(\varepsilon_1\varepsilon_3+\varepsilon_2\varepsilon_4)}.$$

でしたが、円盤では高々２点しか境界がないことを保障していました。しかし、円盤領域に限らずほとんどの図形で領域を十分に細かくすれば、左右の一方と上下の一方の高々２点しか境界に乗らなくすることができるので円盤同様にここでも

$$\tau \leq \frac{\varepsilon_1 \varepsilon_2 h^2}{2(\varepsilon_1 + \varepsilon_2)}.$$

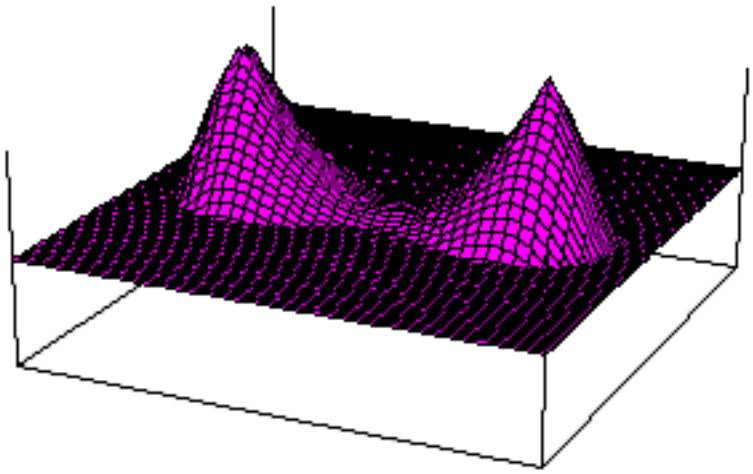
を採用することが出来ます。しかし、最小εを求めるアルゴリズムでは、円左下領域の場合には点の左点と下点が境界であるか判定しましたが、今度のひょうたん型では左下領域に限っても、左右と下の３方向について境界であるか考える必要があります。

7.4　初期値の設定

初期値は円盤のようにキレイにはいきませんが、ここではとんがった角のような山が二つ出るような初期値にしたいと思います。

$$\sqrt{(x^2 + y^2 + a^2)^2 - 4a^2x^2} < b^2$$

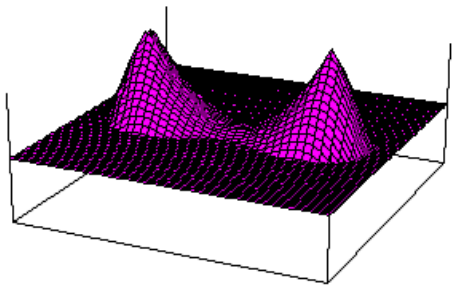
という式なので、b^2 を移項してから全体の符号を変えると先にいったような形になります。



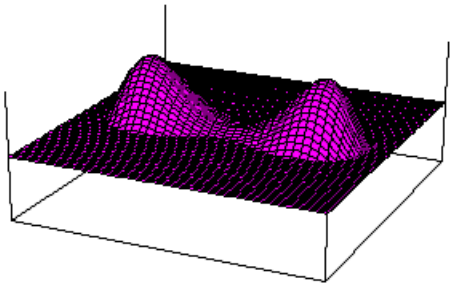
初期値

第8章　実験結果

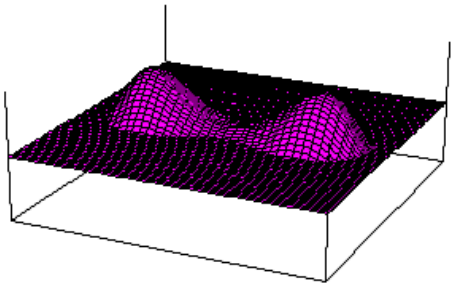
N=40 (正方形領域 80 等分)



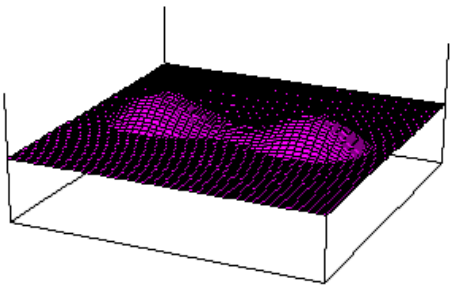
t=0



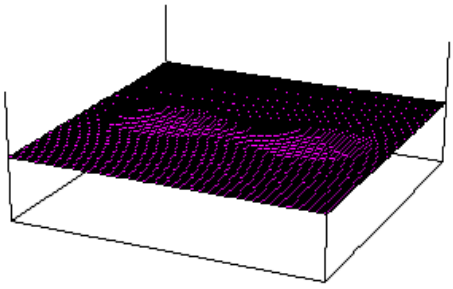
t=0.001



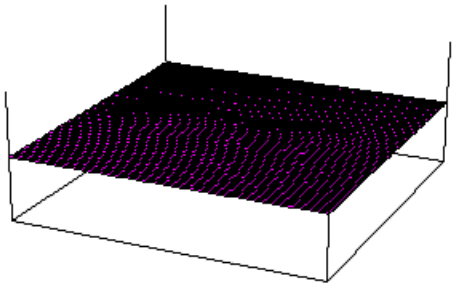
t=0.002



t=0.005



t=0.01



t=0.02

第9章　プログラムリスト

S-W 近似による Cassini の橙形領域の熱方程式を解くプログラム本文です。

散々取り上げてきた $(x^2+y^2+a^2)^2 = 4a^2x^2+b^4$ の式のように a を使いたかったのですが、円盤領域の境界値関数ですでに使ってしまっていたために a の代わりに a を縦に伸ばしたような d を使いました。

```
/*
 * heat2d-e-sw-h.c -- solve the heat equation in a two dimensional square region
 * by explicit finite difference method. (version 2.0)
 *
 *      To compile this program  ->  ccmg heat2d-e-sw-h.c
 */

#include <stdio.h>
#include <math.h>

/* to use matrix, new_matrix() */
#include <matrix.h>

/* to use GLSC */
#define G_DOUBLE
#include <glsc.h>


int main()
{
    int             N, i, j, n, skip, nMax;
    double          d,b;
    double          h, lambda, tau, Tmax, t, c, dt ,yps1,yps2,ypsr,minyps,maxtau;
    double          x,r,hl,hr,hu,hd;
    double          xi,xil,xir,yj,yjd,yju;
    double          ul,ur,uu,ud;
    matrix          u, newu;
    double          f(double, double, double, double);
    double          pita(double,double);
```

```
double          a(double, double);

printf("N: "); scanf("%d",&N);
N=2*N;

h = 1.0 / N;
r=0.5;
yps1=1.0;
yps2=1.0;
maxtau=1.0;
minyps=1.0;
ypsr=1.0e-10;

d=0.3;
b=1.05*d;

 /****************最小εと最大τを求める ****************/
  for (i = 1;i<N/2; i++){
    xi =r-i*h;
    xil=r-(i-1)*h;
    xir=r-(i+1)*h;

    for (j = 1;j<N/2; j++){
      yj=r-j*h;
      if(pita(xi,yj)<b*b-ypsr){
        if(pita(xil,yj)>b*b-ypsr){

          x=sqrt(d*d-yj*yj+sqrt(b*b*b*b-4*d*d*yj*yj));

          hl=x-xi;
          yps1=hl/h;

          if(yps1<minyps)
            minyps=yps1;


          x=sqrt(-d*d-xi*xi+sqrt(b*b*b*b+4*d*d*xi*xi));
          hd=x-yj;

          if(hd>h){
            hd=h;
          }
```

```
        yps2=hd/h;

        if(maxtau>yps1*yps2*h*h/(2.0*(yps1+yps2)))
          maxtau=yps1*yps2*h*h/(2.0*(yps1+yps2));
        }
        if(pita(xir,yj)>b*b-ypsr){

          x=sqrt(d*d-yj*yj-sqrt(b*b*b*b-4*d*d*yj*yj));

          hr=xi-x;
          yps1=hr/h;

          if(yps1<minyps)
            minyps=yps1;


          x=sqrt(-d*d-xi*xi+sqrt(b*b*b*b+4*d*d*xi*xi));
          hd=x-yj;

          if(hd>h){
            hd=h;
          }

        yps2=hd/h;

        if(maxtau>yps1*yps2*h*h/(2.0*(yps1+yps2)))
          maxtau=yps1*yps2*h*h/(2.0*(yps1+yps2));
        }
      }
    }
  }
  printf("εの最小値=%g\n",minyps);

if ((u = new_matrix(N + 1, N + 1)) == NULL) {
    fprintf(stderr, "配列 u を確保できませんでした。");
    exit(1);
}
if ((newu = new_matrix(N + 1, N + 1)) == NULL) {
    fprintf(stderr, "配列 newu を確保できませんでした。");
    exit(1);
}
printf("Tmax: "); scanf("%lf", &Tmax);
```

```
printf("τ (≦%g ): ", maxtau);
scanf("%lf", &tau);


lambda = 2.0*tau/(h*h);
printf("λ=%g になりました。\n", lambda);

printf("Δ t: ");
scanf("%lf",  &dt);
skip = rint(dt / tau);
if (skip == 0) {
  printf("Δ t が小さすぎるので、Δ t=τ とします。\n");
  skip = 1;
}
dt = skip * tau;

g_init("Meta", 250.0, 160.0);
g_device(G_BOTH);
g_def_scale(0, 0.0, 2.0, 0.0, 2.0, 30.0, 70.0, 100.0, 72.0);
g_def_line(0, G_BLACK, 0, G_LINE_SOLID);
g_sel_scale(0);

for (i = 0; i <= N; i++)
    for (j = 0; j <= N; j++)
        u[i][j] = f(i*h, j*h, d, b);

g_cls();
g_hidden2(1.0, 1.0, 0.4, -1.0, 1.0, 5.0, 25.0, 20.0, 20.0, 20.0,
    150.0, 100.0, u, N + 1, N + 1,
    1, G_SIDE_NONE, 2, 1);

nMax = rint(Tmax / tau);

for (n = 1; n <= nMax; n++) {
  /**********************左***********************/
  for (i = 1; i < N/2; i++){
    xi =r-i*h;                          /* u[i][j] の x 座標 */
    xil=r-(i-1)*h;                      /* u[i][j] の左の点の x 座標*/
    xir=r-(i+1)*h;

    /**********************左下**********************/
        for (j = 1; j < N/2; j++){
          yj=r-j*h;                     /* u[i][j] の y 座標 */
```

```
  yjd=r-(j-1)*h;                /* u[i][j] の下の y 座標 */

  if(pita(xi,yj) < b*b-ypsr){
    if(pita(xil,yj) > b*b-ypsr){
      x=sqrt(d*d-yj*yj+sqrt(b*b*b*b-4*d*d*yj*yj));
      hl=x-xi;
      ul=a(r-x,j*h);
    }
    else{
      hl=h;
      ul=u[i-1][j];
    }
  }
  if(pita(xi,yj) < b*b-ypsr){
    if(pita(xir,yj) > b*b-ypsr){
      x=sqrt(d*d-yj*yj-sqrt(b*b*b*b-4*d*d*yj*yj));
      hr=xi-x;
      ur=a(r-x,j*h);
    }
    else{
      hr=h;
      ur=u[i+1][j];
    }
}
if(pita(xi,yj) < b*b-ypsr){
  if(pita(xi,yjd) > b*b-ypsr){
    x=sqrt(-d*d-xi*xi+sqrt(b*b*b*b+4*d*d*xi*xi));
    hd=x-yj;
    ud=a(i*h,r-x);
  }
  else{
    hd=h;
    ud=u[i][j-1];
  }
  hu=h;
  uu=u[i][j+1];

  newu[i][j]=u[i][j]+tau*(2.0*((ur-u[i][j])/hr-(u[i][j]-ul)/hl)/(hr+hl)+
                          2.0*((uu-u[i][j])/hu-(u[i][j]-ud)/hd)/(hu+hd));
}
else{
  newu[i][j] = 0.0;
}
```

```
  }

  /**********************左上**********************/
for (j = N/2; j < N; j++){
  yj=j*h-r;
  yju=(j+1)*h-r;
  if(pita(xi,yj) < b*b-ypsr){
    if(pita(xil,yj) > b*b-ypsr){
      x=sqrt(d*d-yj*yj+sqrt(b*b*b*b-4*d*d*yj*yj));
      hl=x-xi;
      ul=a(r-x,j*h);
    }
    else{
      hl=h;
      ul=u[i-1][j];
    }
  }

  if(pita(xi,yj) < b*b-ypsr){
    if(pita(xir,yj) > b*b-ypsr){
      x=sqrt(d*d-yj*yj-sqrt(b*b*b*b-4*d*d*yj*yj));
      hr=x;
      ur=a(r-x,j*h);
    }
    else{
      hr=h;
      ur=u[i+1][j];
    }
  }
  if(pita(xi,yj) < b*b-ypsr){
    if(pita(xi,yju) > b*b-ypsr){
      x=sqrt(-d*d-xi*xi+sqrt(b*b*b*b+4*d*d*xi*xi));
      hu=x-yj;
      uu=a(i*h,r+x);
    }
    else{
      hu=h;
      uu=u[i][j+1];
    }
    hd=h;
    ud=u[i][j-1];

    newu[i][j]=u[i][j]+tau*(2.0*((ur-u[i][j])/hr-(u[i][j]-ul)/hl)/(hr+hl)+
```

```
                                    2.0*((uu-u[i][j])/hu-(u[i][j]-ud)/hd)/(hu+hd));
      }
      else  newu[i][j]=0.0;
    }
}
/**********************右**********************/
for (i = N/2; i < N; i++){
  xi=i*h-r;
  xir=(i+1)*h-r;
  xil=(i-1)*h-r;

  /**********************右下**************************/
    for (j = 1; j < N/2; j++){
      yj=r-j*h;
      yjd=r-(j-1)*h;
      if(pita(xi,yj) < b*b-ypsr){
        if(pita(xir,yj) > b*b-ypsr){
          x=sqrt(d*d-yj*yj+sqrt(b*b*b*b-4*d*d*yj*yj));
          hr=x-xi;
          ur=a(r+x,j*h);
        }
        else{
          hr=h;
          ur=u[i+1][j];
        }
      }
      if(pita(xi,yj )< b*b-ypsr){
        if(pita(xil,yj) > b*b-ypsr){
          x=sqrt(d*d-yj*yj-sqrt(b*b*b*b-4*d*d*yj*yj));
          hl=xi-x;
          ul=a(r+x,j*h);
        }
        else{
        hl=h;
        ul=u[i-1][j];
        }
      }
      if(pita( xi,yj) < b*b-ypsr){
        if(pita( xi, yjd ) > b*b-ypsr){
          x=sqrt(-d*d-xi*xi+sqrt(b*b*b*b+4*d*d*xi*xi));
          hd=x-yj;
          ud=a(i*h,r-x);
        }
```

```
    else{
      hd=h;
      ud=u[i][j-1];
    }
    hu=h;
    uu=u[i][j+1];

    newu[i][j]=u[i][j]+tau*(2.0*((ur-u[i][j])/hr-(u[i][j]-ul)/hl)/(hr+hl)+
                           2.0*((uu-u[i][j])/hu-(u[i][j]-ud)/hd)/(hu+hd));
  }
  else  newu[i][j] = 0.0;
}
/************************右上***************************/
for (j = N/2; j < N; j++){
  yj=j*h-r;
  yju=(j+1)*h-r;
  if(pita(xi,yj) < b*b-ypsr){
    if(pita(xir,yj) > b*b-ypsr){
      x=sqrt(d*d-yj*yj+sqrt(b*b*b*b-4*d*d*yj*yj));

      hr=x-xi;
      ur=a(r+x,j*h);
    }
    else{
      hr=h;
      ur=u[i+1][j];
    }
  }

    if(pita(xi,yj) < b*b-ypsr){
      if(pita(xil,yj) > b*b-ypsr){
        x=sqrt(d*d-yj*yj-sqrt(b*b*b*b-4*d*d*yj*yj));
        hl=xi-x;
        ul=a(r+x,j*h);
      }
      else{
        hl=h;
        ul=u[i-1][j];
      }
    }

  if(pita(xi,yj) < b*b-ypsr){
    if(pita(xi,yju) > b*b-ypsr){
```

```
                x=sqrt(-d*d-xi*xi+sqrt(b*b*b*b+4*d*d*xi*xi));
                hu=x-yj;
                uu=a(i*h,x+r);
              }
              else{
                hu=h;
                uu=u[i][j+1];
              }
              hd=h;
              ud=u[i][j-1];

              newu[i][j]=u[i][j]+tau*(2.0*((ur-u[i][j])/hr-(u[i][j]-ul)/hl)/(hr+hl)+
                                      2.0*((uu-u[i][j])/hu-(u[i][j]-ud)/hd)/(hu+hd));
            }
            else  newu[i][j]=0.0;
          }
      }
        for (i = 0; i <= N; i++)
            for (j = 0; j <= N; j++)
                u[i][j] = newu[i][j];
        if (n % skip == 0){
            g_cls();
            g_hidden2(1.0, 1.0, 0.4, -1.0, 1.0, 5.0, 25.0, 20.0, 20.0, 20.0,
                150.0, 100.0, u, N + 1, N + 1,
                1, G_SIDE_NONE, 2, 1);
        }
        t = tau * n;
    }
    /* マウスでクリックされるのを待つ */
    g_sleep(-1.0);
    /* ウィンドウを消す */
    g_term();
    return 0;
}

double f(double x, double y,double d,double b){
  double r=0.5;
  if(sqrt(((x-r)*(x-r)+(y-r)*(y-r)+d*d)*((x-r)*(x-r)+(y-r)*(y-r)+d*d)-4*d*d*(x-r)*(x-r)) < b*b-1.
    return 15*(b*b-sqrt(((x-r)*(x-r)+(y-r)*(y-r)+d*d)*((x-r)*(x-r)+(y-r)*(y-r)+d*d)-4*d*d*(x-r)*(

  else  return 0;
}
```

```
double pita(double x,double y){
  double d=0.3;
  double b=1.05*d;
  return sqrt((x*x+y*y+d*d)*(x*x+y*y+d*d)-4*d*d*x*x);
}

double a(double x, double y){
  return 0;
}
```

第10章　付録

10.1　使用したソフト

ここでは、このレポートを作成するにあたって使用したソフトを紹介していきます。すべて、無料で手に入るフリーソフトです。

10.1.1　Inkscape

ベクター曲線（どこまで拡大しても線が滑らかな線）を簡単に作成することが出来る。円を書く場合に非常にきれいにかける。ここでは使用しなかったが、簡単に星やベジエ曲線、カリグラフィックも書けます。

入手先：http://www.altech-ads.com/product/10002198.htm

10.1.2　方眼紙メーカー

本来の目的とは違いますが、格子点を手軽に手に入れることが出来ます。

入手先：http://www.vector.co.jp/soft/win95/writing/se030865.html

10.1.3　キャプチャソフト

上で作成したグラフなどをキャプチャーして画像データにして保存する。Windows は標準でキャプチャー機能があるので、特にこだわりがなければそれを使用してください。

10.1.4　EPS-conv

LaTeX の得意な EPS への変換をドラッグ&ドロップで簡単に行えます。

入手先：http://www.vector.co.jp/soft/win95/art/se176316.html

10.2　参考文献

10.2.1　数値解析入門 [増訂版]　山本哲郎著