2007 年度 卒業研究レポート

S-W 近似によって様々な領域の熱方程式を解く

明治大学 理工学部 数学科
久保田　祥史

2008 年 3 月 5 日

目 次

序

私は2006年度に卒業された金子祐司さんの卒業研究のプログラムを改良し、例えそれがどのような領域の熱方程式であっても、簡単にS-W近似でシミュレートできるようなプログラムを組むことを目標にして、研究してきました。このレポートを見て興味を持てたなら、レポートにのっていない領域も試しに出力してみて、楽しんでくれたら嬉しいです。

第1章　S-W近似によって円盤領域を解く

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」と言います。以下、これをS-W近似と呼びます。通常、差分法は等間隔格子点を用いて行われます。しかし、これは領域が長方形の場合には問題ありませんが、円盤領域等には合いません。よって適当な写像を用いて長方形領域に変換してから、差分法を行ないます。(円盤領域では、極座標変換を行なう。)しかし、このS-W近似ではそのままの領域に対して差分法を行なうことが出来ます。
ここでは求める領域をΩ(R^2の有界領域)とし、熱方程式の初期値境界値問題を以下の様に置きます。

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

1.2　求める領域を差分格子で覆う

分かり易い例として、Ωを円盤領域

$$\Omega = \{(x,y); (x-x_0)^2 + (y-y_0)^2 < r^2\}$$

とします。この領域を図1.1のように、差分格子で覆います。

このとき、x方向の分割数をN_x、y方向の分割数をN_y、x方向の格子点の間隔をh_x、y方向の格子点の間隔をh_y、$x_i = xmin + ih_x$ $(0 \leq i \leq N_x)$、$y_j = ymin + jh_y$ $(0 \leq j \leq N_y)$、時間の刻み幅を$\tau > 0$、として置きます。

1.3　格子点の領域内外の判断

円盤領域$\Omega = \{(x,y); (x-x_0)^2 + (y-y_0)^2 < r^2\}$に対して、任意の格子点が領域の内外かを判断するための関数$F(x,y) = r^2 - ((x-x_0)^2 + (y-y_0)^2)$を置き、ある格子点$P = (x_i, y_j)$に対して

$$\begin{cases} F(x_i,y_j) > 0 & \text{のとき領域内部} \\ F(x_i,y_j) = 0 & \text{のとき境界上} \\ F(x_i,y_j) < 0 & \text{のとき領域外部} \end{cases}$$

として、その格子点Pが領域の内か外、または境界上にあると判断します。

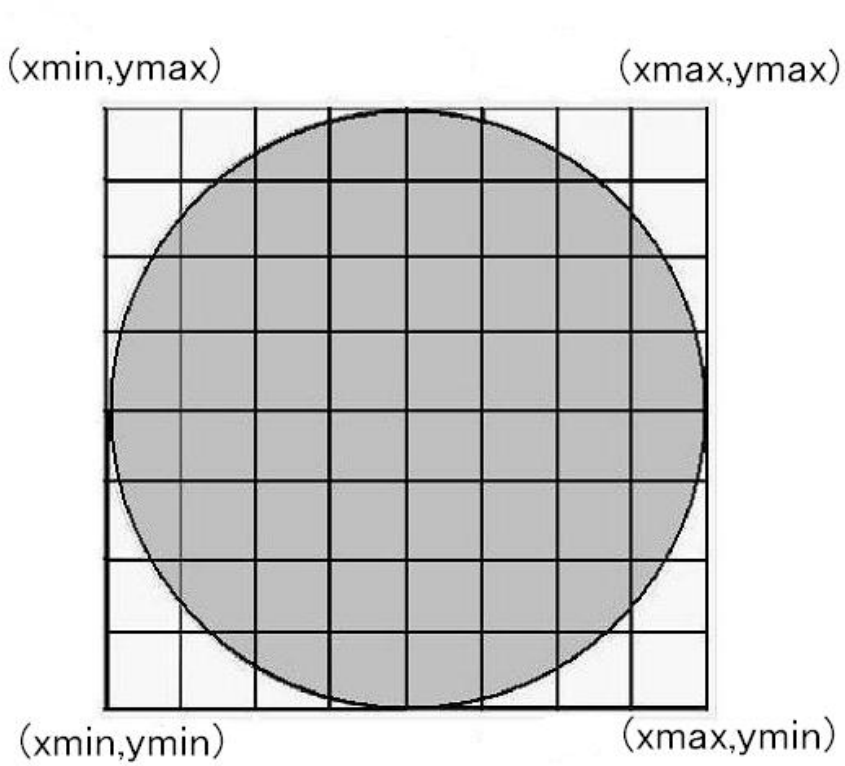


図 1.1: 円盤を格子で覆う

1.4 境界上の不等間隔格子点を取る

等間隔の格子線によって分けられた領域内のある格子点 $P = (x, y)$ に対して、その左右上下の等間隔格子点をそれぞれ

$$P_w = (x_w, y)\ \ P_e = (x_e, y),\ \ P_n = (x, y_n),\ \ P_s = (x, y_s)$$

として置きます。　そのとき、等間隔格子点が領域外に出てしまう時は、図 1.2 のようにその境界上に新しく不等間隔格子点を取り、その点を新たに P_w, P_e, P_n, P_s とします。また、その格子点の間隔をそれぞれ

$$h_w = x - x_w = \varepsilon_w h_x,\ \ h_e = x_e - x = \varepsilon_e h_x$$

$$h_n = y_n - y = \varepsilon_n h_y,\ \ h_s = y - y_s = \varepsilon_s h_y$$

$$(0 < \varepsilon_w, \varepsilon_e, \varepsilon_n, \varepsilon_s \leq 1)$$

として取ります。

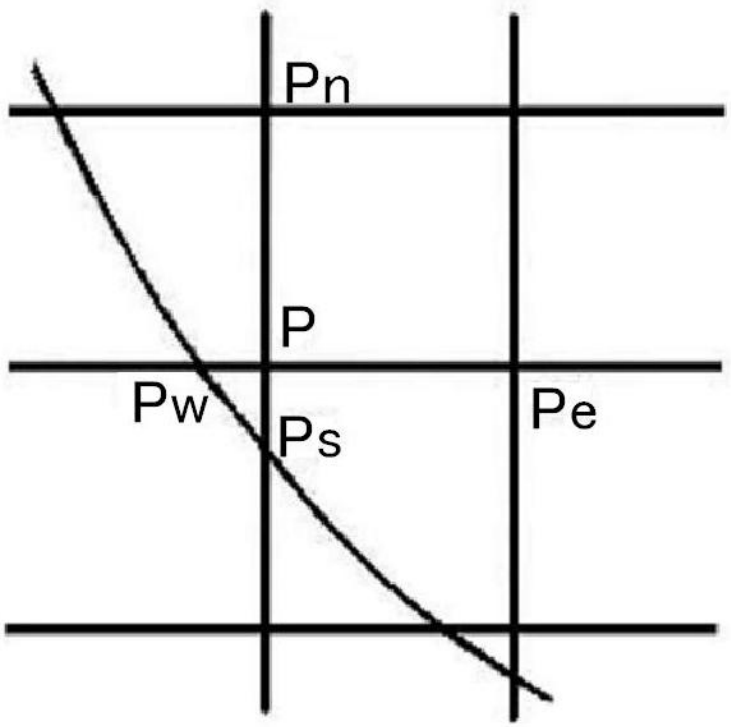


図 1.2: 境界上の座標

1.5 境界の座標の計算

境界がどの格子点の間に存在しているか分かっただけではまだ計算に使うことが出来ません。ここから具体的に座標の値を計算する必要があります。

円盤領域

$$(x - x_0)^2 + (y - y_0)^2 < r^2$$

では、x, y はそれぞれ

$$x = x_0 \pm \sqrt{r^2 - (y - y_0)^2}$$

$$y = y_0 \pm \sqrt{r^2 - (x - x_0)^2}$$

となります。

よって、図 1.2 の場合、P の座標が (x, y) なので

P_w の座標は　$P_w = (x_0 - \sqrt{r^2 - (y - y_0)^2}, y)$

P_s の座標は　$P_s = (x, y_0 - \sqrt{r^2 - (x - x_0)^2})$

P_e の座標は　$P_e = (x_e, y)$

P_n の座標は　$P_n = (x, y_n)$

となります。

1.6　通常の差分法とS-W近似の差分方程式

まず通常の差分法では、Laplacianを

$$\Delta u \backsimeq \frac{1}{h^2}(u_w + u_e + u_n + u_s - 4u)$$

と近似することにより

$$u_{ij}^{n+1} \backsimeq \frac{\tau}{h^2}(u_{i+1j}^n + u_{i-1j}^n + u_{ij+1}^n + u_{ij-1}^n - 4u_{ij}^n) + u_{ij}^n$$

という差分方程式になります。

次に、S-W近似では、式はそれぞれ

$$\Delta u \backsimeq \frac{2}{h_w + h_e}\left(\frac{u_e - u}{h_e} - \frac{u - u_w}{h_w}\right) + \frac{2}{h_n + h_s}\left(\frac{u_n - u}{h_n} - \frac{u - u_s}{h_s}\right),$$

$$u_{ij}^{n+1} \backsimeq u_{ij}^n + \frac{2\tau}{h_w + h_e}\left(\frac{u_e^n - u_{ij}^n}{h_e} - \frac{u_{ij}^n - u_w^n}{h_w}\right) + \frac{2\tau}{h_n + h_s}\left(\frac{u_n^n - u_{ij}^n}{h_n} - \frac{u_{ij}^n - u_s^n}{h_s}\right)$$

となります。これは、$h_w = h_e = h_n = h_s = h$ のとき、通常の差分法の差分方程式に帰結します。

1.7　差分スキームの安定性条件

通常の差分法で長方形領域を解いたとき、τ が

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

を取ると安定します (参考文献 [3])。しかし、円盤領域の場合では不等間隔格子点が存在し、それに伴い h より小さい h_w, h_e, h_n, h_s が発生します。よってここでは、S-W近似における安定性条件を求めます。まず

$$\Delta u \backsimeq \frac{2}{h_w + h_e}\left(\frac{u_e - u}{h_e} - \frac{u - u_w}{h_w}\right) + \frac{2}{h_n + h_s}\left(\frac{u_n - u}{h_n} - \frac{u - u_s}{h_s}\right)$$

に　$h_w = \varepsilon_w hx,\quad h_e = \varepsilon_e hx,\quad h_n = \varepsilon_n hy,\quad h_s = \varepsilon_s hy$　を代入します。すると

$$\Delta u \backsimeq \frac{2}{\varepsilon_w h_x + \varepsilon_e h_x}\left(\frac{u_e - u}{\varepsilon_e h_x} - \frac{u - u_w}{\varepsilon_w h_x}\right) + \frac{2}{\varepsilon_n h_y + \varepsilon_s h_y}\left(\frac{u_n - u}{\varepsilon_n h_y} - \frac{u - u_s}{\varepsilon_s h_y}\right)$$

となり、これを変形すると

$$\Delta u \backsimeq \frac{2(\varepsilon_w u_e + \varepsilon_e u_w)}{h_x^2 \varepsilon_w \varepsilon_e(\varepsilon_w + \varepsilon_e)} + \frac{2(\varepsilon_s u_n + \varepsilon_n u_s)}{h_y^2 \varepsilon_s \varepsilon_n(\varepsilon_s + \varepsilon_n)} - \frac{2(\varepsilon_w \varepsilon_e h_x^2 + \varepsilon_s \varepsilon_n h_y^2)}{\varepsilon_w \varepsilon_e \varepsilon_s \varepsilon_n h_x^2 h_y^2} u$$

となります。

ここで

$$u_t = \Delta u, \quad u_t \backsimeq \frac{u^{n+1} - u^n}{\tau}$$

より、u^{n+1} について解きます。また、ここまでの u は u^n としても表せるので、u^n として書きます。

$$u^{n+1} \backsimeq \frac{2\tau(\varepsilon_w u_e + \varepsilon_e u_w)}{h_x^2 \varepsilon_w \varepsilon_e (\varepsilon_w + \varepsilon_e)} + \frac{2\tau(\varepsilon_s u_n + \varepsilon_n u_s)}{h_y^2 \varepsilon_s \varepsilon_n (\varepsilon_s + \varepsilon_n)} + \Big(1 - \frac{2\tau(\varepsilon_w \varepsilon_e h_x^2 + \varepsilon_s \varepsilon_n h_y^2)}{\varepsilon_w \varepsilon_e \varepsilon_s \varepsilon_n h_x^2 h_y^2}\Big) u$$

となります。

過去の卒研より、差分方程式の右辺の係数が正であることが安定の為の条件です。よって安定の為に u^n の係数が正でなければならないので、

$$1 \geq \frac{2\tau(\varepsilon_w \varepsilon_e h_x^2 + \varepsilon_s \varepsilon_n h_y^2)}{\varepsilon_w \varepsilon_e \varepsilon_s \varepsilon_n h_x^2 h_y^2}$$

τ について整理すると

$$\tau \leq \frac{\varepsilon_w \varepsilon_e \varepsilon_n \varepsilon_s h_x^2 h_y^2}{2(\varepsilon_w \varepsilon_e h_x^2 + \varepsilon_n \varepsilon_s h_y^2)}$$

と表すことができます。

これは $\varepsilon_w,\ \varepsilon_e\ \ \varepsilon_n\ \ \varepsilon_s \to 1$ のとき、

$$\frac{\varepsilon_w \varepsilon_e \varepsilon_n \varepsilon_s h_x^2 h_y^2}{2(\varepsilon_w \varepsilon_e h_x^2 + \varepsilon_n \varepsilon_s h_y^2)} \to \frac{h^2}{4}$$

となります。

$$\varepsilon = \min\{\varepsilon_w, \varepsilon_e, \varepsilon_n, \varepsilon_s\}$$

とおいておきます。

1.8　丸め誤差

図 1.3 のような境界上の点を、PC が丸め誤差のせいで領域の内部だと判断することがあります。よってその対策をプログラムに組み込んでおく必要があります。

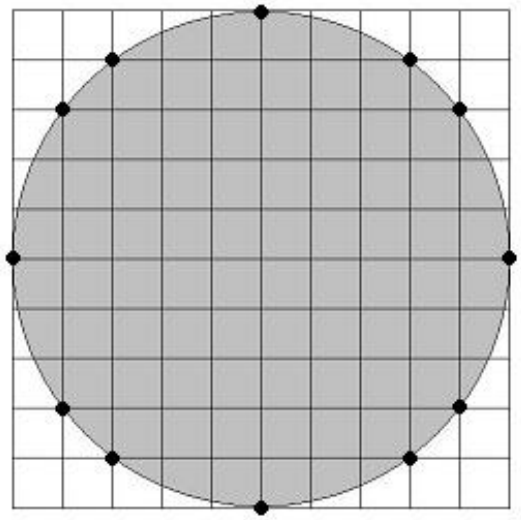


図 1.3: 境界上の格子点

何の対策もしなければ、その等間隔格子点上を領域の内部だと勘違いをして ε を極端に小さい値として返します。*double* 型では、だいたい 10^{-14}〜10^{-15} ぐらいです。この対策として、$\varepsilon_r = 10^{-13}$〜10^{-14} というものを置き、これを用いて格子点の領域の内外を判断します。

具体的にいうと、[1.3 格子点の領域内外の判断] を

$$\begin{cases} F(x_i, y_j) > \varepsilon_r & \text{のとき領域内部} \\ |F(x_i, y_j)| \leq \varepsilon_r & \text{のとき境界上} \\ F(x_i, y_j) < -\varepsilon_r & \text{のとき領域外部} \end{cases}$$

と置き換えます。

またこの誤差は、領域の形や N_x, N_y の大小によって多少変化します。なので、もし ε が ε_r を下回るとき、そのことを出力するようにプログラムを組み込んでおきます。

第2章　円盤領域のソースプログラム

2.1　初期値の設定

初期値とは、最初の温度分布 $f(x,y)=u(x,y,0)$ のことです。以下のプログラムで与えている初期値のグラフはお椀型です。

具体的な式は $f(x,y)=r^2-\sqrt{(x-x_0)^2+(y-y_0)^2}$ となります。

2.2　プログラムの説明

領域 Ω によってプログラムを変化させるわけですが、基本的に変化させる場所は、プログラム前半の/* 領域の内部・境界・外部・の判定用 */ から /* ここまで */ の間です。

F(x,y) は領域 Ω を定める条件を『 $\sim>0$ 』の形に書いたときの、『 $\sim$ 』の部分をプログラムして下さい。

実例：$(x-x_0)^2+(y-y_0)^2<r^2$ を $r^2-((x-x_0)^2+(y-y_0)^2)>0$ として

```
double F(double x, double y, double x0, double y0, double r){
  return r*r-((x-x0)*(x-x0)+(y-y0)*(y-y0));
}
```

とプログラムします。また領域 Ω の方程式を x,y について解きます。

W(y) はその方程式の x の左方を表す式をプログラムして下さい。

E(y) はその方程式の x の右方を表す式をプログラムして下さい。

N(x) はその方程式の y の上方を表す式をプログラムして下さい。

S(x) はその方程式の y の下方を表す式をプログラムして下さい。

実例：$x=x_0\pm\sqrt{r^2-(y-y_0)^2}, y=y_0\pm\sqrt{r^2-(x-x_0)^2}$ よって、

W(y)$=x_0-\sqrt{r^2-(y-y_0)^2}$,E(y)$=x_0+\sqrt{r^2-(y-y_0)^2}$,

N(x)$=y_0+\sqrt{r^2-(x-x_0)^2}$,S(x)$=y_0-\sqrt{r^2-(x-x_0)^2}$

f(x,y) は初期値です。

A が何かというと、xmin, xmax, ymin, ymax (x,y の範囲) や r の値を大きくしていくと、初期値の値が定められたウィンドウの大きさをオーバーしてしまうので、A を用いて、その値を縮小します。

distance は描画領域の中心から視点までの距離です。これが r の値より小さかったり近いと、うまく出力できません。

2.3 soturonE.c

次のようにしてコンパイルして実行します。

```
knoppix$ ccmg soturonE.c
knoppix$ ./soturonE

入力例
 xmin=-5
 xmax=5
 ymin=-5
 ymax=5
 Nx=100
 Ny=100
 x0=0
 y0=0
 r=4
 A=5
 distance=60
```

```
1  /*
2   * sotsuronE.c --- 円盤領域での熱方程式を SW 近似で解く
3   */
4
5
6  #include <stdio.h>
7  #include <math.h>
8
9  /* to use matrix, new_matrix() */
10 #include <matrix.h>
11
12 /* to use GLSC */
13 #define G_DOUBLE
14 #include <glsc.h>
15
16
17 /* 領域の内部・境界・外部の判定用 */
18 double F(double x, double y, double x0, double y0, double r){
19   return r*r-((x-x0)*(x-x0)+(y-y0)*(y-y0));
20 }
21
22 /* 東側の境界 */
23 double E(double y, double x0, double y0, double r){
24   return x0+sqrt(r*r-(y-y0)*(y-y0));
25 }
26
27 /* 西側の境界 */
28 double W(double y, double x0, double y0, double r){
```

```
29      return x0-sqrt(r*r-(y-y0)*(y-y0));
30    }
31
32    /* 北側の境界 */
33    double N(double x, double x0, double y0, double r){
34      return y0+sqrt(r*r-(x-x0)*(x-x0));
35    }
36
37    /* 南側の境界 */
38    double S(double x, double x0, double y0, double r){
39      return y0-sqrt(r*r-(x-x0)*(x-x0));
40    }
41
42    /* 初期条件 */
43    double f(double x, double y, double x0, double y0, double r){
44      return r*r - ((x-x0)*(x-x0)+(y-y0)*(y-y0));
45    }
46    /* ここまで */
47
48    double min(double x, double y){
49      if(x < y){
50      return x;
51      }
52      else return y;
53    }
54
55    int main()
56    {
57      int    Nx, Ny, i, j, n, skip, nMax;
58      double hx, hy, lambda, tau, Tmax, t, dt, ew, es, ee, en, er, mine, maxtau;
59      double x, y, hw, he, hn, hs;
60      double xi, xiw, xie, yj, yjs, yjn;
61      double uw,ue,un,us;
62      double x0, y0, r, A, xmin, xmax, ymin, ymax, distance;
63      matrix u, newu;
64
65
66      /* 領域のｘ軸、ｙ軸の長さ */
67      printf("xmin=  "); scanf("%lf",&xmin);
68      printf("xmax=  "); scanf("%lf",&xmax);
69      printf("ymin=  "); scanf("%lf",&ymin);
70      printf("ymax=  "); scanf("%lf",&ymax);
71
72      /* 分割数 */
73      printf("Nx=  "); scanf("%d",&Nx);
74      printf("Ny=  "); scanf("%d",&Ny);
75
76      /* 円の中心 */
77      printf("x0=  "); scanf("%lf",&x0);
78      printf("y0=  "); scanf("%lf",&y0);
79
```

```
80    /* 円の半径 */
81    printf("r=  "); scanf("%lf",&r);
82
83    /* 縮小倍率 */
84    printf("A=  "); scanf("%lf",&A);
85
86    /* 視点の距離 */
87    printf("distance=  "); scanf("%lf",&distance);
88
89    /* 格子間の長さ */
90    hx = (xmax - xmin)/Nx;
91    hy = (ymax - ymin)/Ny;
92    printf("hx=%g, hy=%g\n", hx, hy);
93
94    /* 誤差 */
95    er= 1.0e-14;
96
97    mine= 1.0;
98    maxtau= 1.0;
99
100   /****************最小εと最大τを求める *****************/
101
102   for (i = 0;i <= Nx; i++){
103     xi = xmin+i*hx;
104     xiw = xmin+(i-1)*hx;
105     xie = xmin+(i+1)*hx;
106
107     for (j = 0;j <= Ny; j++){
108       yj = ymin+j*hy;
109       yjs = ymin+(j-1)*hy;
110       yjn = ymin+(j+1)*hy;
111
112       if(F(xi, yj, x0, y0, r) >= er){
113 /* WEST */
114         if(F(xiw, yj, x0, y0, r) >= er){
115           hw= hx;
116           ew= 1.0;
117         }
118         else if(fabs(F(xiw, yj, x0, y0, r)) < er){
119           ew= 1.0;
120           hw= hx;
121         }
122         else{
123           hw= xi - W(yj, x0, y0, r);
124           ew= hw/hx;
125         }
126 /* SOUTH */
127         if(F(xi, yjs, x0, y0, r) >= er){
128           hs= hy;
129           es= 1.0;
130         }
```

```
131           else if(fabs(F(xi, yjs, x0, y0, r)) < er){
132             hs= hy;
133             es= 1.0;
134           }
135           else{
136             hs= yj - S(xi, x0, y0, r);
137             es= hs/hy;
138           }
139   /* EAST */
140           if(F(xie, yj, x0, y0, r) >= er){
141             he= hx;
142             ee= 1.0;
143           }
144           else if(fabs(F(xie, yj, x0, y0, r)) < er){
145             ee= 1.0;
146             he= hx;
147           }
148           else{
149             he= E(yj, x0, y0, r) - xi;
150             ee= he/hx;
151           }
152   /* NORTH */
153           if(F(xi, yjn, x0, y0, r) >= er){
154             en= 1.0;
155             hn= hy;
156           }
157           else if(fabs(F(xi, yjn, x0, y0, r)) < er){
158             en= 1.0;
159             hn= hy;
160           }
161           else{
162             hn= N(xi, x0, y0, r) - yj;
163             en= hn/hy;
164           }
165         }
166         else {
167           ew = ee = en = es = 1.0;
168           hw = he = hx;
169           hn = hs = hy;
170         }
171
172         mine = min(mine, min(min(ew,ee),min(en,es)));
173         maxtau = min(maxtau,
174                  ew*es*ee*en*hx*hx*hy*hy/(2.0*(hx*hx*ew*ee+hy*hy*es*en)));
175       }
176     }
177
178     printf("εの最小値= %g\n", mine);
179     printf("τの最大値=%g\n", maxtau);
180     if(mine < er){
181       printf("mine=%g: *******εが小さすぎます********\n", mine);
```

```
182     }
183
184     if ((u = new_matrix(Nx + 1, Ny + 1)) == NULL) {
185       fprintf(stderr, "配列 u を確保できませんでした。");
186       exit(1);
187     }
188     if ((newu = new_matrix(Nx + 1, Ny + 1)) == NULL) {
189       fprintf(stderr, "配列 newu を確保できませんでした。");
190       exit(1);
191     }
192
193     printf("Tmax=  "); scanf("%lf", &Tmax);
194     printf("τ (≦ %g )==  ", maxtau); scanf("%lf", &tau);
195
196     lambda = tau/(hx*hx) + tau/(hy*hy);
197     printf("λ= %g になりました。\n", lambda);
198     printf("Δ t=  "); scanf("%lf",  &dt);
199
200     skip = rint(dt / tau);
201     if (skip == 0) {
202       printf("Δ tが小さすぎるので、Δ t=  τ  とします。\n");
203       skip = 1;
204       dt = skip * tau;
205     }
206
207     g_init("En", 250.0, 170.0);
208     g_device(G_BOTH);
209     g_def_scale(0, xmin, xmax, ymin, ymax, 70.0, 70.0, 100.0, 100.0);
210     g_def_line(0, G_BLACK, 0, G_LINE_SOLID);
211     g_sel_scale(0);
212
213     for (i = 0; i <= Nx; i++)
214       for (j = 0; j <= Ny; j++)
215         u[i][j] = f(xi, yj, x0, y0, r)/A;
216
217     nMax = rint(Tmax / tau);
218
219     for (n = 0; n <= nMax; n++) {
220
221       /**********************領域************************/
222
223       for (i = 0; i <= Nx; i++){
224         xi = xmin+i*hx;
225         xiw = xmin+(i-1)*hx;
226         xie = xmin+(i+1)*hx;
227
228         for (j = 0; j <= Ny; j++){
229           yj = ymin+j*hy;
230           yjn = ymin+(j+1)*hy;
231           yjs = ymin+(j-1)*hy;
232
```

```
233           if(F(xi, yj, x0, y0, r) >= er){
234   /* WEST */
235            if(F(xiw, yj, x0, y0, r) >= er){
236              hw= hx;
237              uw= u[i-1][j];
238            }
239            else if(fabs(F(xiw, yj, x0, y0, r)) < er){
240              hw= hx;
241              uw= u[i-1][j];
242            }
243            else{
244              hw= xi - W(yj, x0, y0, r);
245              uw= 0;
246            }
247   /* EAST */
248            if(F(xie, yj, x0, y0, r) >= er){
249              he=hx;
250              ue=u[i+1][j];
251            }
252            else if(fabs(F(xie, yj, x0, y0, r)) < er){
253              he=hx;
254              ue=u[i+1][j];
255            }
256            else{
257              he= E(yj, x0, y0, r) - xi;
258              ue= 0;
259            }
260   /* NORTH */
261            if(F(xi, yjn, x0, y0, r) >= er){
262              hn=hy;
263              un=u[i][j+1];
264            }
265            else if(fabs(F(xi, yjn, x0, y0, r)) < er){
266              hn=hy;
267              un=u[i][j+1];
268            }
269            else{
270              hn= N(xi, x0, y0, r) - yj;
271              un= 0;
272            }
273   /* SOUTH */
274            if(F(xi, yjs, x0, y0, r) >= er){
275              hs=hy;
276              us=u[i][j-1];
277            }
278            else if(fabs(F(xi, yjs, x0, y0, r)) < er){
279              hs=hy;
280              us=u[i][j-1];
281            }
282            else{
283              hs= yj - S(xi, x0, y0, r);
```

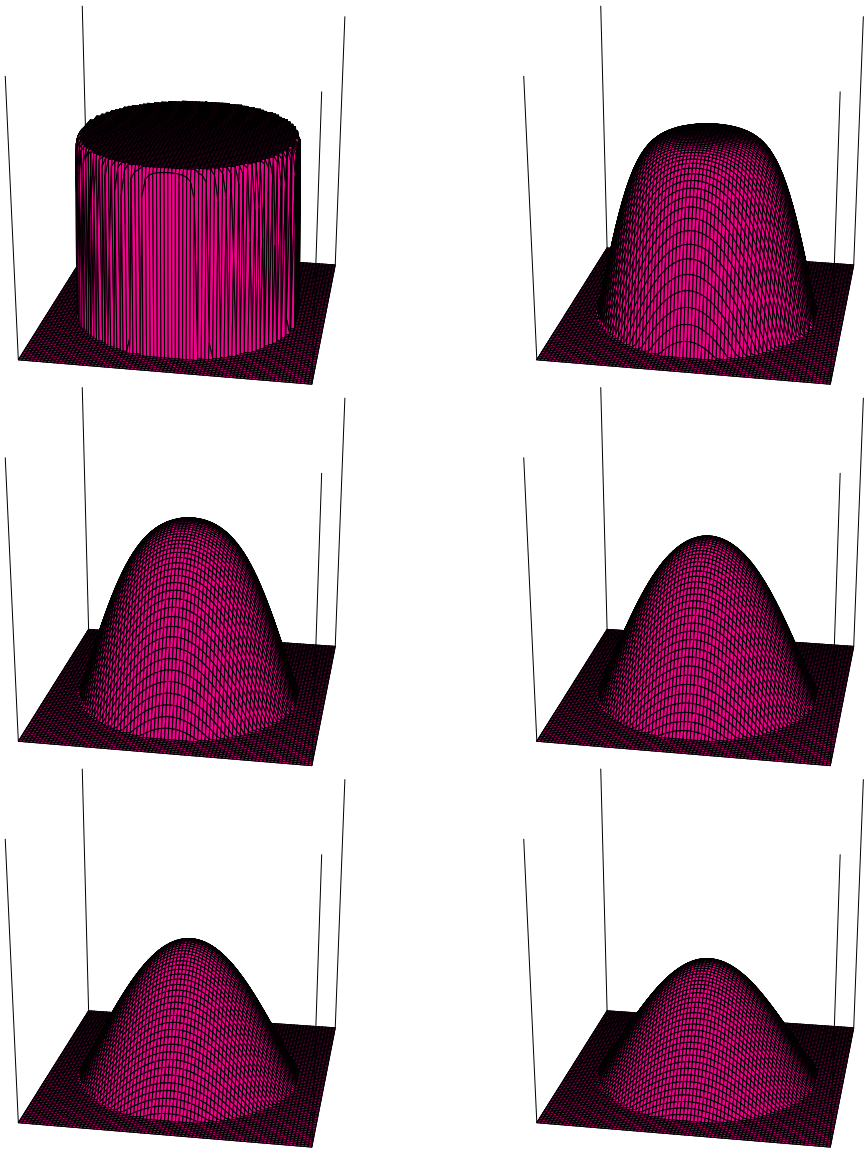
```
284               us= 0;
285             }
286           }
287           else{
288             u[i][j]= 0;
289             hw= he= hx;
290             hn= hs= hy;
291             uw= ue= un= us= 0;
292           }
293
294           newu[i][j] = u[i][j]
295             + 2.0*tau*((ue-u[i][j])/he - (u[i][j]-uw)/hw) /(he+hw)
296             + 2.0*tau*((un-u[i][j])/hn - (u[i][j]-us)/hs) /(hn+hs);
297
298         }
299       }
300
301       for (i = 0; i <= Nx; i++)
302         for (j = 0; j <= Ny; j++)
303           u[i][j] = newu[i][j];
304           if (n % skip == 0){
305             g_cls();
306             g_hidden2(xmax-xmin, ymax-ymin, 10.0, 0.0, -10.0, distance,
307                       10.0, 20.0, 50.0, 50.0, 150.0, 100.0,
308                       u, Nx+1, Ny+1, 1, G_SIDE_NONE, 1, 1);
309           }
310           t = tau * n;
311     }
312     /* マウスでクリックされるのを待つ */
313     g_sleep(-1.0);
314     /* ウィンドウを消す */
315     g_term();
316     return 0;
317   }
```

2.4　実験結果

$$xmin = 0, xmax = 10, ymin = 0, ymax = 10, x_0 = 5, y_0 = 5$$

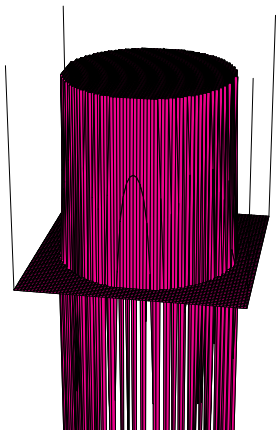
$$N_x = N_y = 80, r = 4, A = 5, distance = 60$$

$t = 0$〜3　　0.5 刻み

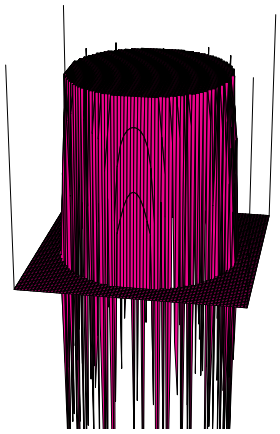


2.5　安定性条件を守らなかった場合

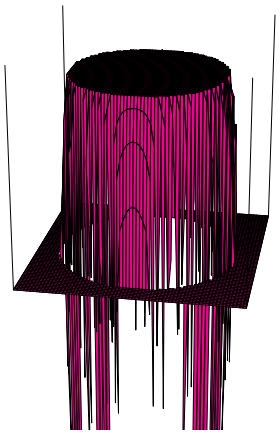
もし安定性条件を守らずにシミュレートした場合、境界部分から激しく振動していき、それが全体に広がっていきます。以下の図はその崩壊の様子です。今回の τ は安定性条件の約 10 倍を取りました。



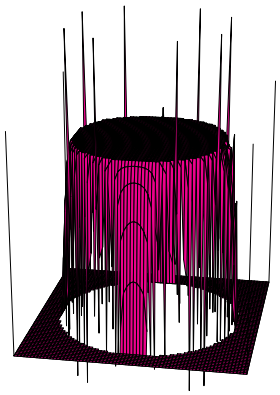
t=0



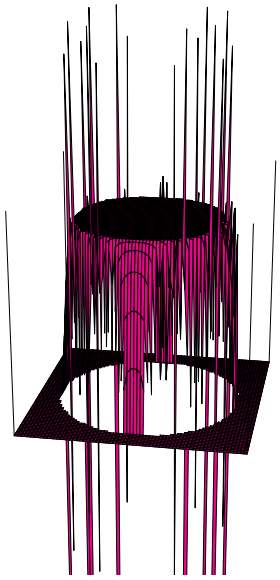
t=0.001



t=0.002



t=0.003



t=0.004

第3章　他の領域のプログラム

この章では円以外の領域のプログラムを紹介します。

3.1　楕円

楕円の領域は

$$\Omega = \{(x, y); \frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} < r^2\}.$$

また、楕円の方程式を x,y について解くと、それぞれ

$$x = x_0 \pm a\sqrt{1 - \frac{(y - y_0)^2}{b^2}}$$

$$y = y_0 \pm b\sqrt{1 - \frac{(x - x_0)^2}{a^2}}$$

となります。
プログラムは以下のものになります。

```
1   /*
2    * sotsuronD.c --- 楕円領域での熱方程式を SW 近似で解く
3    */
4
5   #include <stdio.h>
6   #include <math.h>
7
8   /* to use matrix, new_matrix() */
9   #include <matrix.h>
10
11  /* to use GLSC */
12  #define G_DOUBLE
13  #include <glsc.h>
14
15
16  /* 領域の内部・境界・外部の判定用 */
17  double F(double x, double y, double a, double b, double x0, double y0){
18    return a*a*b*b - (b*b*(x-x0)*(x-x0) + a*a*(y-y0)*(y-y0));
19  }
20
21  /* 東側の境界 */
22  double E(double y, double a, double b, double x0, double y0){
23    return x0 + a* sqrt(1 - (y-y0)*(y-y0)/(b*b));
```

```
24  }
25
26  /* 西側の境界 */
27  double W(double y, double a, double b, double x0, double y0){
28    return x0 - a* sqrt(1 - (y-y0)*(y-y0)/(b*b));
29  }
30
31  /* 北側の境界 */
32  double N(double x, double a, double b, double x0, double y0){
33    return y0 + b* sqrt(1 - (x-x0)*(x-x0)/(a*a));
34  }
35
36  /* 南側の境界 */
37  double S(double x, double a, double b, double x0, double y0){
38    return y0 - b* sqrt(1 - (x-x0)*(x-x0)/(a*a));
39  }
40
41  /* 初期条件 */
42  double f(double x, double y, double a, double b, double x0, double y0){
43    return a*a*b*b - (b*b*(x-x0)*(x-x0) + a*a*(y-y0)*(y-y0));
44  }
45  /* ここまで */
46
47  double min(double x, double y){
48    if(x < y){
49    return x;
50    }
51    else return y;
52  }
53
54
55  int main()
56  {
57    int    Nx, Ny, i, j, n, skip, nMax;
58    double hx, hy, lambda, tau, Tmax, t, dt, ew, es, ee, en, er, mine, maxtau;
59    double x, y, hw, he, hn, hs;
60    double xi, xiw, xie, yj, yjs, yjn;
61    double uw,ue,un,us;
62    double a, b, xmin, xmax, ymin, ymax, x0, y0, A, distance;
63    matrix u, newu;
64
65
66    /* 領域の x 軸、y 軸の長さ */
67    printf("xmin=  "); scanf("%lf",&xmin);
68    printf("xmax=  "); scanf("%lf",&xmax);
69    printf("ymin=  "); scanf("%lf",&ymin);
70    printf("ymax=  "); scanf("%lf",&ymax);
71
72    /* 分割数 */
73    printf("Nx=  "); scanf("%d",&Nx);
74    printf("Ny=  "); scanf("%d",&Ny);
```

```
75
76      /* 円の中心 */
77      printf("x0=  "); scanf("%lf",&x0);
78      printf("y0=  "); scanf("%lf",&y0);
79
80      /* 円の半径 */
81      printf("a=  "); scanf("%lf",&a);
82      printf("b=  "); scanf("%lf",&b);
83
84      /* 倍率 */
85      printf("A=  "); scanf("%lf",&A);
86
87      /* 距離 */
88      printf("distance=  "); scanf("%lf",&distance);
89
90      /* 格子間の長さ */
91      hx = (xmax-xmin)/Nx;
92      hy = (ymax-ymin)/Ny;
93      printf("hx=%g, hy=%g\n", hx, hy);
94
95      /* 誤差 */
96      er= 1.0e-13;
97
98      mine= 1.0;
99      maxtau= 1.0;
100
101     /****************最小εと最大τを求める *****************/
102
103     for (i = 0;i <= Nx; i++){
104       xi = xmin+i*hx;
105       xiw = xmin+(i-1)*hx;
106       xie = xmin+(i+1)*hx;
107
108       for (j = 0;j <= Ny; j++){
109         yj = ymin+j*hy;
110         yjs = ymin+(j-1)*hy;
111         yjn = ymin+(j+1)*hy;
112
113         if(F(xi, yj, a, b, x0, y0) >= er){
114   /* WEST */
115           if(F(xiw, yj, a, b, x0, y0) >= er){
116             hw= hx;
117             ew= 1.0;
118           }
119           else if(fabs(F(xiw, yj, a, b, x0, y0)) < er){
120             ew= 1.0;
121             hw= hx;
122           }
123           else{
124             hw= xi - W(yj, a, b, x0, y0);
125             ew= hw/hx;
```

```
126           }
127   /* SOUTH */
128           if(F(xi, yjs, a, b, x0, y0) >= er){
129             hs= hy;
130             es= 1.0;
131           }
132           else if(fabs(F(xi, yjs, a, b, x0, y0)) < er){
133             hs= hy;
134             es= 1.0;
135           }
136           else{
137             hs= yj - S(xi, a, b, x0, y0);
138             es= hs/hy;
139           }
140   /* EAST */
141           if(F(xie, yj, a, b, x0, y0) >= er){
142             he= hx;
143             ee= 1.0;
144           }
145           else if(fabs(F(xie, yj, a, b, x0, y0)) < er){
146             ee= 1.0;
147             he= hx;
148           }
149           else {
150             he= E(yj, a, b, x0, y0) - xi;
151             ee= he/hx;
152           }
153   /* NORTH */
154           if(F(xi, yjn, a, b, x0, y0) >= er){
155             en= 1.0;
156             hn= hy;
157           }
158           else if(fabs(F(xi, yjn, a, b, x0, y0)) < er){
159             en= 1.0;
160             hn= hy;
161           }
162           else{
163             hn= N(xi, a, b, x0, y0) - yj;
164             en= hn/hy;
165           }
166         }
167         else {
168           ew = ee = en = es = 1.0;
169           hw = he = hx;
170           hn = hs = hy;
171         }
172
173         mine = min(mine, min(min(ew,ee),min(en,es)));
174         maxtau = min(maxtau,
175               ew*es*ee*en*hx*hx*hy*hy/(2.0*(hx*hx*ew*ee+hy*hy*es*en)));
176       }
```

```
177     }
178
179     printf("εの最小値= %g\n", mine);
180     printf("τの最大値=%g\n", maxtau);
181     if(mine < er){
182       printf("mine=%g: *******εが小さすぎます********\n", mine);
183     }
184
185     if ((u = new_matrix(Nx + 1, Ny + 1)) == NULL) {
186       fprintf(stderr, "配列 u を確保できませんでした。");
187       exit(1);
188     }
189     if ((newu = new_matrix(Nx + 1, Ny + 1)) == NULL) {
190       fprintf(stderr, "配列 newu を確保できませんでした。");
191       exit(1);
192     }
193
194     printf("Tmax=  "); scanf("%lf", &Tmax);
195     printf("τ (≦ %g )==  ", maxtau); scanf("%lf", &tau);
196
197     lambda = tau/(hx*hx) + tau/(hy*hy);
198     printf("λ= %g になりました。\n", lambda);
199
200     printf("Δ t=  "); scanf("%lf",  &dt);
201
202     skip = rint(dt / tau);
203     if (skip == 0) {
204       printf("Δ tが小さすぎるので、Δ t=  τ  とします。\n");
205       skip = 1;
206       dt = skip * tau;
207     }
208
209     g_init("Meta", 250.0, 170.0);
210     g_device(G_BOTH);
211     g_def_scale(0, xmin, xmax, ymin, ymax, 100.0, 100.0, 100.0, 100.0);
212     g_def_line(0, G_BLACK, 0, G_LINE_SOLID);
213     g_sel_scale(0);
214
215     for (i = 0; i <= Nx; i++)
216       for (j = 0; j <= Ny; j++)
217         u[i][j] = f(xi, yj, a, b, x0, y0)/A;
218
219     nMax = rint(Tmax / tau);
220
221     for (n = 0; n <= nMax; n++) {
222
223       /**********************領域************************/
224
225       for (i = 0; i <= Nx; i++){
226         xi = xmin+i*hx;
227         xiw = xmin+(i-1)*hx;
```

```
228         xie = xmin+(i+1)*hx;
229
230         for (j = 0; j <= Ny; j++){
231           yj = ymin+j*hy;
232           yjn = ymin+(j+1)*hy;
233           yjs = ymin+(j-1)*hy;
234
235           if(F(xi, yj, a, b, x0, y0) >= er){
236   /* WEST */
237            if(F(xiw, yj, a, b, x0, y0) >= er){
238              hw= hx;
239              uw= u[i-1][j];
240            }
241            else if(fabs(F(xiw, yj, a, b, x0, y0)) < er){
242              hw= hx;
243              uw= u[i-1][j];
244            }
245            else{
246              hw= xi - W(yj, a, b, x0, y0);
247              uw= 0;
248            }
249   /* EAST */
250            if(F(xie, yj, a, b, x0, y0) >= er){
251              he=hx;
252              ue=u[i+1][j];
253            }
254            else if(fabs(F(xie, yj, a, b, x0, y0)) < er){
255              he=hx;
256              ue=u[i+1][j];
257            }
258            else{
259              he= E(yj, a, b, x0, y0) - xi;
260              ue= 0;
261            }
262   /* NORTH */
263            if(F(xi, yjn, a, b, x0, y0) >= er){
264              hn=hy;
265              un=u[i][j+1];
266            }
267            else if(fabs(F(xi, yjn, a, b, x0, y0)) < er){
268              hn=hy;
269              un=u[i][j+1];
270            }
271            else{
272              hn= N(xi, a, b, x0, y0) - yj;
273              un= 0;
274            }
275   /* SOUTH */
276            if(F(xi, yjs, a, b, x0, y0) >= er){
277              hs=hy;
278              us=u[i][j-1];
```

```
279             }
280             else if(fabs(F(xi, yjs, a, b, x0, y0)) < er){
281               hs=hy;
282               us=u[i][j-1];
283             }
284             else{
285               hs= yj - S(xi, a, b, x0, y0);
286               us= 0;
287             }
288           }
289           else{
290             u[i][j]= 0;
291             hw= he= hx;
292             hn= hs= hy;
293             uw= ue= un= us= 0;
294           }
295
296           newu[i][j] = u[i][j]
297                       + 2.0*tau*((ue-u[i][j])/he - (u[i][j]-uw)/hw) /(he+hw)
298                       + 2.0*tau*((un-u[i][j])/hn - (u[i][j]-us)/hs )/(hn+hs);
299
300         }
301       }
302
303       for (i = 0; i <= Nx; i++)
304         for (j = 0; j <= Ny; j++)
305           u[i][j] = newu[i][j];
306           if (n % skip == 0){
307             g_cls();
308             g_hidden2(xmax-xmin, ymax-ymin, 10.0, 0.0, -10.0, distance,
309                       10.0, 20.0, 50.0, 50.0, 150.0, 100.0, u, Nx+1, Ny+1,
310                       1, G_SIDE_NONE, 1, 1);
311           }
312           t = tau * n;
313     }
314     /* マウスでクリックされるのを待つ */
315     g_sleep(-1.0);
316     /* ウィンドウを消す */
317     g_term();
318     return 0;
319   }
```

3.2 円環領域

円環の領域は

$$\Omega = \{(x, y); r^2 < (x - x_0)^2 + (y - y_0)^2 < R^2\}$$

と表します。図にすると

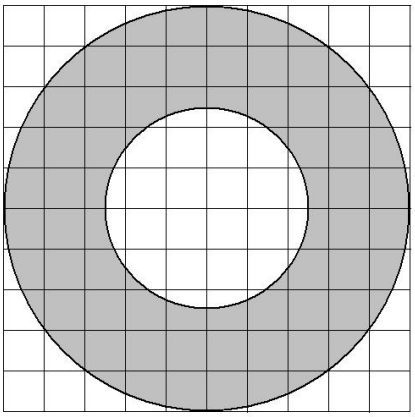


図 3.1: 円環の領域

となります。

また、図 3.1 からも分かるように x,y が

$$x = \begin{cases} x_0 \pm \sqrt{r^2 - (y - y_0)^2} \\ \\ x_0 \pm \sqrt{R^2 - (y - y_0)^2} \end{cases}$$

$$y = \begin{cases} b \pm \sqrt{r^2 - (x - a)^2} \\ \\ b \pm \sqrt{R^2 - (x - a)^2} \end{cases}$$

とそれぞれ４つ式があります。そこでどうするかというと、図 3.2 のしるしをつけている所を関数 N(x) が対応しているので、

$$N(x) = \begin{cases} y_0 + \sqrt{R^2 - (x - x_0)^2} & (y > b \text{ のとき}) \\ \\ y_0 - \sqrt{r^2 - (x - x_0)^2} & (y \leq b \text{ のとき}) \end{cases}$$

と場合分けします。

他の関数 $W(y), E(y), S(x)$ も同様に行ないます。

プログラムは以下のものになります。

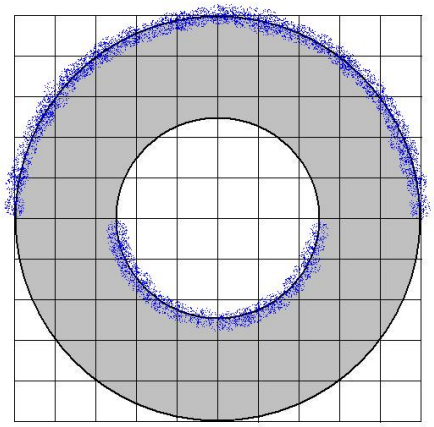


図 3.2: N(x) が対応する所

```
1  /*
2   * sotsuronENKAN.c --- 円環領域での熱方程式を SW 近似で解く
3   */
4
5
6  #include <stdio.h>
7  #include <math.h>
8
9  /* to use matrix, new_matrix() */
10 #include <matrix.h>
11
12 /* to use GLSC */
13 #define G_DOUBLE
14 #include <glsc.h>
15
16
17 /* 領域の内部・境界・外部の判定用 */
18 double F(double x, double y, double x0, double y0, double R, double r)
19 {
20   double t = hypot(x-x0, y-y0);
21   return - (t - r) * (t - R);
22 }
23
24 /* 東側の境界 */
25 double E(double x, double y, double x0, double y0, double R, double r){
26   if(x > x0){
27     return x0 + sqrt(R*R - (y-y0)*(y-y0));
28   }
29   else return x0 - sqrt(r*r - (y-y0)*(y-y0));
30 }
31
32 /* 西側の境界 */
33 double W(double x, double y, double x0, double y0, double R, double r){
34   if(x > x0){
35     return x0 + sqrt(r*r - (y-y0)*(y-y0));
```

```
36    }
37    else return x0 - sqrt(R*R - (y-y0)*(y-y0));
38  }
39
40  /* 北側の境界 */
41  double N(double x, double y, double x0, double y0, double R, double r){
42    if(y > y0){
43      return y0 + sqrt(R*R - (x-x0)*(x-x0));
44    }
45    else return y0 - sqrt(r*r - (x-x0)*(x-x0));
46  }
47
48  /* 南側の境界 */
49  double S(double x, double y, double x0, double y0, double R, double r){
50    if(y > y0){
51      return y0 + sqrt(r*r - (x-x0)*(x-x0));
52    }
53    else return y0 - sqrt(R*R - (x-x0)*(x-x0));
54  }
55
56  /* 初期条件 */
57  double f(double x, double y, double x0, double y0, double R){
58    return R*R - ((x-x0)*(x-x0) + (y-y0)*(y-y0));
59  }
60  /* ここまで */
61
62  double min(double x, double y){
63    if(x < y){
64    return x;
65    }
66    else return y;
67  }
68
69
70  int main()
71  {
72    int    Nx, Ny, i, j, n, skip, nMax;
73    double hx, hy, lambda, tau, Tmax, t, dt;
74    double ew, es, ee, en, er, mine, maxtau;
75    double x, y, hw, he, hn, hs;
76    double xi, xiw, xie, yj, yjs, yjn;
77    double uw,ue,un,us;
78    double x0, y0, r, R, xmin, xmax, ymin, ymax, A, distance;
79    matrix u, newu;
80
81
82    /* 領域の x 軸、y 軸の長さ */
83    printf("xmin=  "); scanf("%lf",&xmin);
84    printf("xmax=  "); scanf("%lf",&xmax);
85    printf("ymin=  "); scanf("%lf",&ymin);
86    printf("ymax=  "); scanf("%lf",&ymax);
```

```
87
88      /* 分割数 */
89      printf("Nx=  "); scanf("%d",&Nx);
90      printf("Ny=  "); scanf("%d",&Ny);
91
92      /* 円の中心 */
93      printf("x0=  "); scanf("%lf",&x0);
94      printf("y0=  "); scanf("%lf",&y0);
95
96      /* 円の半径 */
97      printf("R=  "); scanf("%lf",&R);
98      printf("r=  "); scanf("%lf",&r);
99
100     /* 倍率 */
101     printf("A=  "); scanf("%lf",&A);
102
103     /* 距離 */
104     printf("distance=  "); scanf("%lf",&distance);
105
106     /* 格子間の長さ */
107     hx = (xmax-xmin)/Nx;
108     hy = (ymax-ymin)/Ny;
109     printf("hx=%g, hy=%g\n", hx, hy);
110
111     /* 誤差 */
112     er= 1.0e-14;
113
114     mine= 1.0;
115     maxtau= 1.0;
116
117     /****************最小εと最大τを求める *****************/
118
119     for (i = 0;i <= Nx; i++){
120       xi = xmin+i*hx;
121       xiw = xmin+(i-1)*hx;
122       xie = xmin+(i+1)*hx;
123
124       for (j = 0;j <= Ny; j++){
125         yj = ymin+j*hy;
126         yjs = ymin+(j-1)*hy;
127         yjn = ymin+(j+1)*hy;
128
129         if(F(xi, yj, x0, y0, R, r) >= er){
130   /* WEST */
131           if(F(xiw, yj, x0, y0, R, r) >= er){
132             hw= hx;
133             ew= 1.0;
134           }
135           else if(fabs(F(xiw, yj, x0, y0, R, r)) < er){
136             ew= 1.0;
137             hw= hx;
```

```
138           }
139           else{
140             hw= xi - W(xi, yj, x0, y0, R, r);
141             ew= hw/hx;
142           }
143   /* SOUTH */
144           if(F(xi, yjs, x0, y0, R, r) >= er){
145             hs= hy;
146             es= 1.0;
147           }
148           else if(fabs(F(xi, yjs, x0, y0, R, r)) < er){
149             hs= hy;
150             es= 1.0;
151           }
152           else{
153             hs= yj - S(xi, yj, x0, y0, R, r);
154             es= hs/hy;
155           }
156   /* EAST */
157           if(F(xie, yj, x0, y0, R, r) >= er){
158             he= hx;
159             ee= 1.0;
160           }
161           else if(fabs(F(xie, yj, x0, y0, R, r)) < er){
162             ee= 1.0;
163             he= hx;
164           }
165           else {
166             he= E(xi, yj, x0, y0, R, r) - xi;
167             ee= he/hx;
168           }
169   /* NORTH */
170           if(F(xi, yjn, x0, y0, R, r) >= er){
171             en= 1.0;
172             hn= hy;
173           }
174           else if(fabs(F(xi, yjn, x0, y0, R, r)) < er){
175             en= 1.0;
176             hn= hy;
177           }
178           else{
179             hn= N(xi, yj, x0, y0, R, r) - yj;
180             en= hn/hy;
181           }
182         }
183         else {
184           ew = ee = en = es = 1.0;
185           hw = he = hx;
186           hn = hs = hy;
187         }
188
```

```
189         mine = min(mine, min(min(ew,ee),min(en,es)));
190         maxtau = min(maxtau,
191                 ew*es*ee*en*hx*hx*hy*hy/(2.0*(hx*hx*ew*ee+hy*hy*es*en)));
192       }
193     }
194
195     printf("εの最小値= %g\n", mine);
196     printf("τの最大値=%g\n", maxtau);
197     if(mine < er){
198       printf("mine=%g: *******εが小さすぎます********\n", mine);
199     }
200
201     if ((u = new_matrix(Nx + 1, Ny + 1)) == NULL) {
202       fprintf(stderr, "配列 u を確保できませんでした。");
203       exit(1);
204     }
205     if ((newu = new_matrix(Nx + 1, Ny + 1)) == NULL) {
206       fprintf(stderr, "配列 newu を確保できませんでした。");
207       exit(1);
208     }
209
210     printf("Tmax=  "); scanf("%lf", &Tmax);
211     printf("τ (≦ %g )==  ", maxtau); scanf("%lf", &tau);
212
213     lambda = tau/(hx*hx) + tau/(hy*hy);
214     printf("λ= %g になりました。\n", lambda);
215
216     printf("Δ t=  "); scanf("%lf",  &dt);
217
218     skip = rint(dt / tau);
219     if (skip == 0) {
220       printf("Δ tが小さすぎるので、Δ t=  τ  とします。\n");
221       skip = 1;
222       dt = skip * tau;
223     }
224
225     g_init("Meta", 250.0, 250.0);
226     g_device(G_BOTH);
227     g_def_scale(0, xmin, xmax, ymin, ymax, 100.0, 100.0, 100.0, 100.0);
228     g_def_line(0, G_BLACK, 0, G_LINE_SOLID);
229     g_sel_scale(0);
230
231     /* 初期条件 */
232     for (i = 0; i <= Nx; i++)
233       for (j = 0; j <= Ny; j++)
234         u[i][j] = f(xi, yj, x0, y0, R)/A;
235
236     nMax = rint(Tmax / tau);
237
238     for (n = 0; n <= nMax; n++) {
239
```

```
240       /**********************領域************************/
241
242       for (i = 0; i <= Nx; i++){
243         xi = xmin+i*hx;
244         xiw = xmin+(i-1)*hx;
245         xie = xmin+(i+1)*hx;
246
247         for (j = 0; j <= Ny; j++){
248           yj = ymin+j*hy;
249           yjn = ymin+(j+1)*hy;
250           yjs = ymin+(j-1)*hy;
251
252           if(F(xi, yj, x0, y0, R, r) >= er){
253   /* WEST */
254            if(F(xiw, yj, x0, y0, R, r) >= er){
255              hw= hx;
256              uw= u[i-1][j];
257            }
258            else if(fabs(F(xiw, yj, x0, y0, R, r)) < er){
259              hw= hx;
260              uw= u[i-1][j];
261            }
262            else{
263              hw= xi - W(xi, yj, x0, y0, R, r);
264              uw= 0;
265            }
266   /* EAST */
267            if(F(xie, yj, x0, y0, R, r) >= er){
268              he=hx;
269              ue=u[i+1][j];
270            }
271            else if(fabs(F(xie, yj, x0, y0, R, r)) < er){
272              he=hx;
273              ue=u[i+1][j];
274            }
275            else{
276              he= E(xi, yj, x0, y0, R, r) - xi;
277              ue= 0;
278            }
279   /* NORTH */
280            if(F(xi, yjn, x0, y0, R, r) >= er){
281              hn=hy;
282              un=u[i][j+1];
283            }
284            else if(fabs(F(xi, yjn, x0, y0, R, r)) < er){
285              hn=hy;
286              un=u[i][j+1];
287            }
288            else{
289              hn= N(xi, yj, x0, y0, R, r) - yj;
290              un= 0;
```

```
291             }
292   /* SOUTH */
293             if(F(xi, yjs, x0, y0, R, r) >= er){
294               hs=hy;
295               us=u[i][j-1];
296             }
297             else if(fabs(F(xi, yjs, x0, y0, R, r)) < er){
298               hs=hy;
299               us=u[i][j-1];
300             }
301             else{
302               hs= yj - S(xi, yj, x0, y0, R, r);
303               us= 0;
304             }
305           }
306           else{
307             u[i][j]= 0;
308             hw= he= hx;
309             hn= hs= hy;
310             uw= ue= un= us= 0;
311           }
312
313           newu[i][j] = u[i][j]
314                    + 2.0*tau*((ue-u[i][j])/he - (u[i][j]-uw)/hw) /(he+hw)
315
316         }
317       }
318
319       for (i = 0; i <= Nx; i++)
320         for (j = 0; j <= Ny; j++)
321           u[i][j] = newu[i][j];
322           if (n % skip == 0){
323             g_cls();
324             g_hidden2(xmax-xmin, ymax-ymin, 10.0, 0.0, -10.0, distance,
325                       10.0, 20.0, 50.0, 50.0, 150.0, 150.0, u, Nx+1, Ny+1,
326                       1, G_SIDE_NONE, 1, 1);
327           }
328           t = tau * n;
329     }
330     /* マウスでクリックされるのを待つ */
331     g_sleep(-1.0);
332     /* ウィンドウを消す */
333     g_term();
334     return 0;
335   }
```

3.3 Cassini の oval

Cassini の oval とは、方程式

$$((x-x_0)^2+(y-y_0)^2+a^2)^2 = b^4+4a^2(x-x_0)^2$$

で与えられる曲線です。この曲線は、2 点 $(x_0+a, y_0), (x_0-a, y_0)$ からの距離の積が一定値 b^2 に等しい点の軌跡です。

x, y について解くと、それぞれ、

$$x = x_0 \pm \sqrt{a^2-(y-y_0)^2 \pm \sqrt{b^4-4a^2(y-y_0)^2}},$$

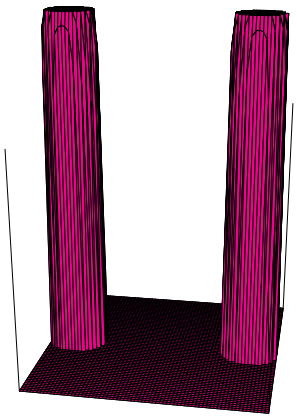
$$y = y_0 \pm \sqrt{-a^2-(x-x_0)^2 \pm \sqrt{b^4+4a^2(x-x_0)^2}}$$

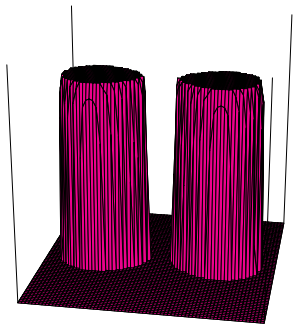
となります。ただし、$\sqrt{\ }$ 内は正でなければならないので、

$$y = y_0 \pm \sqrt{-a^2-(x-x_0)^2 + \sqrt{b^4+4a^2(x-x_0)^2}}$$

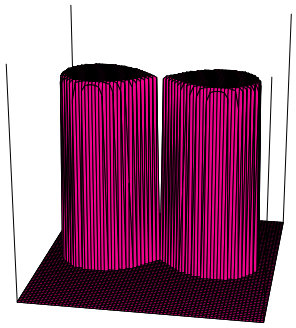
となります。また $Cassini$ の $oval$ は a, b の大小関係で以下のように形を変えます。

$a > b$ のとき　　(a と b の値が大きいほど左寄り)

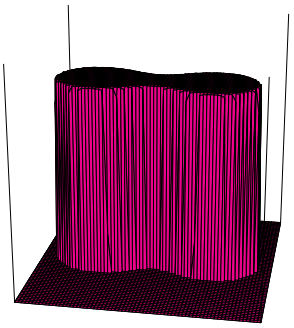


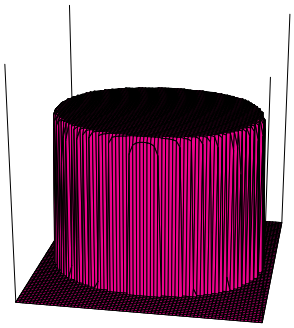


$a = b$ のとき　　(8 の字型)



$a < b$ のとき　　(a と b の値が大きいほど右寄り)





プログラムは以下のものになります。

```
1   /*
2    * sotsuronC.c --- Cassinin の oval での熱方程式を SW 近似で解く
3    */
4
5   #include <stdio.h>
6   #include <math.h>
7
8   /* to use matrix, new_matrix() */
9   #include <matrix.h>
10
11  /* to use GLSC */
12  #define G_DOUBLE
13  #include <glsc.h>
14
15
16  /* 領域の内部・境界・外部の判定用 */
17  double F(double x, double y, double a, double b, double x0, double y0){
18    return b*b
19        - sqrt(((x-x0)*(x-x0)+(y-y0)*(y-y0)+a*a)*((x-x0)*(x-x0)+(y-y0)*(y-y0)+a*a)
20        -4.0*a*a*(x-x0)*(x-x0));
21  }
22
23  /* 東側の境界 */
24  double E(double x, double y, double a, double b, double x0, double y0){
25    if(x <= x0){
26    return x0-sqrt(a*a - (y-y0)*(y-y0) - sqrt(b*b*b*b-4*a*a*(y-y0)*(y-y0)));
27    }
28    else return x0+sqrt(a*a - (y-y0)*(y-y0) + sqrt(b*b*b*b-4*a*a*(y-y0)*(y-y0)));
29  }
30
31  /* 西側の境界 */
32  double W(double x, double y, double a, double b, double x0, double y0){
33    if(x <= x0){
```

```
34    return x0-sqrt(a*a - (y-y0)*(y-y0) + sqrt(b*b*b*b-4*a*a*(y-y0)*(y-y0)));
35    }
36    else return x0+sqrt(a*a - (y-y0)*(y-y0) - sqrt(b*b*b*b-4*a*a*(y-y0)*(y-y0)));
37  }
38
39  /* 北側の境界 */
40  double N(double x, double a, double b, double x0, double y0){
41    return y0+sqrt(-a*a - (x-x0)*(x-x0) + sqrt(b*b*b*b+4*a*a*(x-x0)*(x-x0)));
42  }
43
44  /* 南側の境界 */
45  double S(double x, double a, double b, double x0, double y0){
46    return y0-sqrt(-a*a - (x-x0)*(x-x0) + sqrt(b*b*b*b+4*a*a*(x-x0)*(x-x0)));
47  }
48
49  /* 初期条件 */
50  double f(double x, double y, double a, double b, double x0, double y0){
51    return b*b
52           - sqrt(((x-x0)*(x-x0)+(y-y0)*(y-y0)+a*a)*((x-x0)*(x-x0)
53           +(y-y0)*(y-y0)+a*a) -4.0*a*a*(x-x0)*(x-x0));
54  }
55  /* ここまで */
56
57  double min(double x, double y){
58    if(x < y){
59    return x;
60    }
61    else return y;
62  }
63
64
65  int main()
66  {
67    int    Nx, Ny, i, j, n, skip, nMax;
68    double hx, hy, lambda, tau, Tmax, t, dt, ew, es, ee, en, er, mine, maxtau;
69    double x, y, hw, he, hn, hs;
70    double xi, xiw, xie, yj, yjs, yjn;
71    double uw,ue,un,us;
72    double a, b, A, xmin, xmax, ymin, ymax, x0, y0, distance;
73    matrix u, newu;
74
75
76    /* 領域のｘ軸、ｙ軸の長さ */
77    printf("xmin=  "); scanf("%lf",&xmin);
78    printf("xmax=  "); scanf("%lf",&xmax);
79    printf("ymin=  "); scanf("%lf",&ymin);
80    printf("ymax=  "); scanf("%lf",&ymax);
81
82    /* 分割数 */
83    printf("Nx=  "); scanf("%d",&Nx);
84    printf("Ny=  "); scanf("%d",&Ny);
```

```
85
86      /* 円の中心 */
87      printf("x0=  "); scanf("%lf",&x0);
88      printf("y0=  "); scanf("%lf",&y0);
89
90      /* 円の半径 */
91      printf("a=  "); scanf("%lf",&a);
92      printf("b=  "); scanf("%lf",&b);
93
94      /* 倍率 */
95      printf("A=  "); scanf("%lf",&A);
96
97      /* 距離 */
98      printf("distance=  "); scanf("%lf",&distance);
99
100     /* 格子間の長さ */
101     hx = (xmax-xmin)/Nx;
102     hy = (ymax-ymin)/Ny;
103     printf("hx=%g, hy=%g\n", hx, hy);
104
105     /* 誤差 */
106     er= 1.0e-14;
107
108     mine= 1.0;
109     maxtau= 1.0;
110
111     /****************最小εと最大τを求める *****************/
112
113     for (i = 0;i <= Nx; i++){
114       xi = xmin+i*hx;
115       xiw = xmin+(i-1)*hx;
116       xie = xmin+(i+1)*hx;
117
118       for (j = 0;j <= Ny; j++){
119         yj = ymin+j*hy;
120         yjs = ymin+(j-1)*hy;
121         yjn = ymin+(j+1)*hy;
122
123         if(F(xi, yj, a, b, x0, y0) >= er){
124   /* WEST */
125           if(F(xiw, yj, a, b, x0, y0) >= er){
126             hw= hx;
127             ew= 1.0;
128           }
129           else if(fabs(F(xiw, yj, a, b, x0, y0)) < er){
130             ew= 1.0;
131             hw= hx;
132           }
133           else{
134             hw= xi - W(xi, yj, a, b, x0, y0);
135             ew= hw/hx;
```

```
136           }
137   /* SOUTH */
138           if(F(xi, yjs, a, b, x0, y0) >= er){
139             hs= hy;
140             es= 1.0;
141           }
142           else if(fabs(F(xi, yjs, a, b, x0, y0)) < er){
143             hs= hy;
144             es= 1.0;
145           }
146           else{
147             hs= yj - S(xi, a, b, x0, y0);
148             es= hs/hy;
149           }
150   /* EAST */
151           if(F(xie, yj, a, b, x0, y0) >= er){
152             he= hx;
153             ee= 1.0;
154           }
155           else if(fabs(F(xie, yj, a, b, x0, y0)) < er){
156             ee= 1.0;
157             he= hx;
158           }
159           else{
160             he= E(xi, yj, a, b, x0, y0) - xi;
161             ee= he/hx;
162           }
163   /* NORTH */
164           if(F(xi, yjn, a, b, x0, y0) >= er){
165             en= 1.0;
166             hn= hy;
167           }
168           else if(fabs(F(xi, yjn, a, b, x0, y0)) < er){
169             en= 1.0;
170             hn= hy;
171           }
172           else{
173             hn= N(xi, a, b, x0, y0) - yj;
174             en= hn/hy;
175           }
176         }
177         else {
178           ew = ee = en = es = 1.0;
179           hw = he = hx;
180           hn = hs = hy;
181         }
182
183        mine= min(mine, min(min(ew,ee),min(en,es)));
184        maxtau= min(maxtau,
185                  ew*es*ee*en*hx*hx*hy*hy/(2.0*(hx*hx*ew*ee+hy*hy*es*en)));
186
```

```
187       }
188     }
189
190     printf("εの最小値= %g\n", mine);
191     printf("τの最大値=%g\n", maxtau);
192     if(mine < er){
193       printf("mine=%g: *******εが小さすぎます********\n", mine);
194     }
195
196     if ((u = new_matrix(Nx + 1, Ny + 1)) == NULL) {
197       fprintf(stderr, "配列 u を確保できませんでした。");
198       exit(1);
199     }
200     if ((newu = new_matrix(Nx + 1, Ny + 1)) == NULL) {
201       fprintf(stderr, "配列 newu を確保できませんでした。");
202       exit(1);
203     }
204
205     printf("Tmax=  "); scanf("%lf", &Tmax);
206
207     printf("τ (≦ %g )==  ", maxtau); scanf("%lf", &tau);
208
209     lambda = tau/(hx*hx) + tau/(hy*hy);
210     printf("λ= %g になりました。\n", lambda);
211
212     printf("Δ t=  "); scanf("%lf",  &dt);
213
214     skip = rint(dt / tau);
215     if (skip == 0) {
216       printf("Δ tが小さすぎるので、Δ t=  τ  とします。\n");
217       skip = 1;
218       dt = skip * tau;
219     }
220
221     g_init("Meta", 250.0, 170.0);
222     g_device(G_BOTH);
223     g_def_scale(0, xmin, xmax, ymin, ymax, 100.0, 100.0, 100.0, 100.0);
224     g_def_line(0, G_BLACK, 0, G_LINE_SOLID);
225     g_sel_scale(0);
226
227     for (i = 0; i <= Nx; i++)
228       for (j = 0; j <= Ny; j++)
229         u[i][j] = f(xi, yj, a, b, x0, y0)/A;
230
231     nMax = rint(Tmax / tau);
232
233     for (n = 0; n <= nMax; n++) {
234
235       /**********************領域************************/
236
237       for (i = 0; i <= Nx; i++){
```

```
238         xi = xmin+i*hx;
239         xiw = xmin+(i-1)*hx;
240         xie = xmin+(i+1)*hx;
241
242         for (j = 0; j <= Ny; j++){
243           yj = ymin+j*hy;
244           yjn = ymin+(j+1)*hy;
245           yjs = ymin+(j-1)*hy;
246
247           if(F(xi, yj, a, b, x0, y0) >= er){
248   /* WEST */
249            if(F(xiw, yj, a, b, x0, y0) >= er){
250              hw= hx;
251              uw= u[i-1][j];
252            }
253            else if(fabs(F(xiw, yj, a, b, x0, y0)) < er){
254              hw= hx;
255              uw= u[i-1][j];
256            }
257            else{
258              hw= xi - W(xi, yj, a, b, x0, y0);
259              uw= 0;
260            }
261   /* EAST */
262            if(F(xie, yj, a, b, x0, y0) >= er){
263              he=hx;
264              ue=u[i+1][j];
265            }
266            else if(fabs(F(xie, yj, a, b, x0, y0)) < er){
267              he=hx;
268              ue=u[i+1][j];
269            }
270            else{
271              he= E(xi, yj, a, b, x0, y0) - xi;
272              ue= 0;
273            }
274   /* NORTH */
275            if(F(xi, yjn, a, b, x0, y0) >= er){
276              hn=hy;
277              un=u[i][j+1];
278            }
279            else if(fabs(F(xi, yjn, a, b, x0, y0)) < er){
280              hn=hy;
281              un=u[i][j+1];
282            }
283            else{
284              hn= N(xi, a, b, x0, y0) - yj;
285              un= 0;
286            }
287   /* SOUTH */
288            if(F(xi, yjs, a, b, x0, y0) >= er){
```

```
289               hs=hy;
290               us=u[i][j-1];
291             }
292             else if(fabs(F(xi, yjs, a, b, x0, y0)) < er){
293               hs=hy;
294               us=u[i][j-1];
295             }
296             else{
297               hs= yj - S(xi, a, b, x0, y0);
298               us= 0;
299             }
300           }
301           else{
302             u[i][j]= 0;
303             hw= he= hx;
304             hn= hs= hy;
305             uw= ue= un= us= 0;
306           }
307
308           newu[i][j] = u[i][j]
309                      + 2.0*tau*((ue-u[i][j])/he - (u[i][j]-uw)/hw) /(he+hw)
310                      + 2.0*tau*((un-u[i][j])/hn - (u[i][j]-us)/hs )/(hn+hs);
311
312         }
313       }
314
315       for (i = 0; i <= Nx; i++)
316         for (j = 0; j <= Ny; j++)
317           u[i][j] = newu[i][j];
318           if (n % skip == 0){
319             g_cls();
320             g_hidden2(xmax-xmin, ymax-ymin, 10.0, 0.0, -10.0, distance,
321                       10.0, 20.0, 50.0, 50.0, 150.0, 100.0, u, Nx+1, Ny+1,
322                       1, G_SIDE_NONE, 1, 1);
323           }
324           t = tau * n;
325     }
326     /* マウスでクリックされるのを待つ */
327     g_sleep(-1.0);
328     /* ウィンドウを消す */
329     g_term();
330     return 0;
331   }
```

第4章　付録

4.1　使用したソフト

4.1.1　方眼紙メーカー

本来の用途ではありませんが、これで格子点の図を作りました。
http://www.vecter.co.jp/soft/win95/writing/se030865.html

参考文献

[1] 金子 祐司, 『S-W近似によって様々な領域の熱方程式を解くアルゴリズム』, 2006年度桂田研卒業研究, http://www.math.meiji.ac.jp/~mk/labo/report/open/2006-kaneko.pdf

[2] 山本 哲朗, 数値解析入門 [増訂版], サイエンス社 (2003).

[3] 桂田 祐史, 熱方程式に対する差分法II, http://www.math.meiji.ac.jp/~mk/labo/text/heat-fdm-2.pdf