2004 年度卒業研究レポート

円盤、円柱領域における熱方程式に対する差分法

明治大学　理工学部　数学科
岡田　俊宣

2005 年 2 月 28 日

目 次

第1章　序

1.1　はじめに

私は今回、まず円盤領域における熱方程式を異なる差分スキームを用いて解き、先輩のレポート（遠藤・高木・内藤 [6]) を参考にして数値実験を追試した。後で詳しく述べるが、先輩の予想した安定条件で、陽解法ではとても厳しい結果になった。そこで別の差分スキームで解いた場合はどうなるのか (半陰スキーム、ADI 法)、長方形領域の類推で安定性が予想できるのかを調べた。そして、行列のスペクトル半径から安定性を調べること、さらに円柱領域における差分法について（プログラムも作る）まで取り組むことを目標にしてきた。

1.2　問題

$$u_t(x,y,t) = \triangle u(x,y,t) \qquad ((x,y) \in \Omega, t \in (0,\infty)) \tag{1.1}$$

$$u(x,y,t) = 0 \qquad ((x,y) \in \Gamma, t \in (0,\infty)) \tag{1.2}$$

$$u(x,y,0) = f(x,y) \qquad ((x,y) \in \overline{\Omega}, t = 0) \tag{1.3}$$

$$\Omega = B(0;1) = \{(x,y) \in \mathbf{R}^2; x^2 + y^2 < 1\} \tag{1.4}$$

本来ならまず上記の問題の厳密解を求めるが、ここではそれについては省略する。この問題での厳密解の公式だけ書くと

$$u(r,\theta,t) = \sum_{n=0}^{\infty}\sum_{m=1}^{\infty} e^{-\mu_{m,n}t} J_n(\mu_{m,n}r)(A_{nm}\cos n\theta + B_{nm}\sin n\theta)$$

である。ここで、J_n は n 次の Bessel 関数、 $\mu_{m,n}$ は J_n の正の零点の小さいほうから m 番目の値である。

次章から差分方程式を用いた数値計算法について述べる。

第2章　円盤領域における差分法

2.1　陽解法

陽解法とは、問題 (1.1) のような偏微分方程式を差分近似で置き換えると、ある時刻の解をそれ以前の時刻の解を使って陽に解くことができる、というものである。こうすると、時刻を増しながら次々と解を求めていくことによって、初期値・境界値問題を解くことができる。

2.1.1　差分方程式の導出

問題の式を極座標変換すると、

$$\begin{aligned}
&\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} \qquad (t>0; 0<r<1; 0 \le \theta \le 2\pi),\\
&u(1,\theta,t) = 0 \qquad (t>0; 0\le\theta\le 2\pi),\\
&u(r,\theta,0) = f(r,\theta) \qquad (0\le r\le 1; 0\le\theta\le 2\pi).
\end{aligned}$$

これから述べる計算法はどれも、上式から差分方程式を導出する。今回は陽解法で、領域の格子点を (r_i, θ_j) として各時間ステップごとに近似値を求める。

格子点の分割数として自然数 N_r, N_θ をとり、

$$h_r \overset{\text{def.}}{=} \frac{1}{N_r}, \qquad h_\theta \overset{\text{def.}}{=} \frac{2\pi}{N_\theta},$$

さらに

$$\begin{aligned}
r_i &\overset{\text{def.}}{=} ih_r \qquad (i = 0, 1, \cdots, N_r),\\
\theta_j &\overset{\text{def.}}{=} jh_\theta \qquad (j = 0, 1, \cdots, N_\theta)
\end{aligned}$$

とおく。

次に、$\tau > 0$ を固定して、

$$t_n \overset{\text{def.}}{=} n\tau \qquad (n = 0, 1, 2, \cdots)$$

とする。そして、

$$\lambda_r \stackrel{\text{def.}}{=} \frac{\tau}{h_r^2}, \qquad \lambda_\theta \stackrel{\text{def.}}{=} \frac{\tau}{h_\theta^2}$$

とおく。
目標は、(r_i, θ_j, t_n) における $u_{i,j}^n = u(r_i, \theta_j, t_n)$ の近似値 $U_{i,j}^n$ を求めることである。
　まず、$u_t(r_i, \theta_j, t_n)$ を

$$u_t(r_i, \theta_j, t_n) \fallingdotseq \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\tau}$$

と前進差分近似する。
$u_{rr}(r_i, \theta_j, t_n)$ と $u_{\theta\theta}(r_i, \theta_j, t_n)$ をそれぞれ

$$u_{rr}(r_i, \theta_j, t_n) \fallingdotseq \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{h_r^2},$$

$$u_{\theta\theta}(r_i, \theta_j, t_n) \fallingdotseq \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{h_\theta^2}$$

と 2 階中心差分近似し、$u_r(r_i, \theta_j, t_n)$ を

$$u_r(r_i, \theta_j, t_n) \fallingdotseq \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2h_r}$$

と 1 階中心差分近似すると、差分方程式

$$\frac{U_{i,j}^{n+1} - U_{i,j}^n}{\tau} = \frac{U_{i+1,j}^n - 2U_{i,j}^n + U_{i-1,j}^n}{h_r^2} + \frac{1}{r_i}\frac{U_{i+1,j}^n - U_{i-1,j}^n}{2h_r} + \frac{1}{r_i^2}\frac{U_{i,j+1}^n - 2U_{i,j}^n + U_{i,j-1}^n}{h_\theta^2}$$

を得る。
分母を払って、

$$U_{i,j}^{n+1} - U_{i,j}^n = \lambda_r(U_{i+1,j}^n - 2U_{i,j}^n + U_{i-1,j}^n) + \frac{\lambda_r h_r}{2r_i}(U_{i+1,j}^n - U_{i-1,j}^n) + \frac{\lambda_\theta}{r_i^2}(U_{i,j+1}^n - 2U_{i,j}^n + U_{i,j-1}^n)$$

移項して整理すると、

$$U_{i,j}^{n+1} = [1 - 2\lambda_r - \frac{\lambda_\theta}{r_i^2}]U_{i,j}^n + \lambda_r[(1 + \frac{h_r}{2r_i})U_{i+1,j}^n + (1 - \frac{h_r}{2r_i})U_{i-1,j}^n] + \frac{\lambda_\theta}{r_i^2}(U_{i,j+1}^n + U_{i,j-1}^n) \tag{2.1}$$

$$(i = 1, 2, \cdots, N_r - 1; j = 0, 1, \cdots, N_\theta - 1).$$

ただし、$U_{i,N_\theta}^n = U_{i,0}^n$,$U_{i,-1}^n = U_{i,N\theta-1}^n$ と考える。

一方、境界条件から

$$U_{N_r,j}^{n+1} = f(\cos\theta_j, \sin\theta_j) \qquad (j = 0, 1, \cdots, N_\theta - 1)$$

と書ける。

2.1.2 原点での差分近似

ここで、原点について考える。

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2}$$

を元に戻して

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}.$$

ここで、半径 h_r、中心が原点の円を描く。この円が x 軸、y 軸と交わる点における値は、$u_{1,0}$,$u_{1,\frac{N_\theta}{4}}$,$u_{1,\frac{N_\theta}{2}}$,$u_{1,\frac{3N_\theta}{4}}$、中心の値は $u_{0,0}$ となる。

2 階中心差分近似をして、

$$\frac{\partial^2 u}{\partial x^2}(0,0,n\tau) = \frac{u_{1,0} - 2u_{0,0} + u_{1,\frac{N_\theta}{2}}}{h_r^2} + O(h_r^2),$$

$$\frac{\partial^2 u}{\partial y^2}(0,0,n\tau) = \frac{u_{1,\frac{N_\theta}{4}} - 2u_{0,0} + u_{1,\frac{3N_\theta}{4}}}{h_r^2} + O(h_r^2).$$

よって上式から、

$$\triangle u(0,0,n\tau) = \frac{u_{1,0} + u_{1,\frac{N_\theta}{4}} + u_{1,\frac{N_\theta}{2}} + u_{1,\frac{3N_\theta}{4}} - 4u_{0,0}}{h_r^2} + O(h_r^2)$$

と書ける。N_θ が 4 の倍数ならば、$U_{i,j} = u(r_i, \theta_j)$ として

$$\triangle u(0,0,n\tau) \fallingdotseq \frac{4}{h_r^2}\left[\frac{1}{4}\left(U_{1,0} + U_{1,\frac{N_\theta}{2}} + U_{1,\frac{N_\theta}{4}} + U_{1,\frac{3N_\theta}{4}}\right) - U_{0,0}\right].$$

ところで、Laplacian は座標系の回転に関して不変であるから、任意の j について

$$\triangle u(0,0,n\tau) \fallingdotseq \frac{4}{h_r^2}\left[\frac{1}{4}\left(U_{1,j} + U_{1,\frac{N_\theta}{2}+j} + U_{1,\frac{N_\theta}{4}+j} + U_{1,\frac{3N_\theta}{4}+j}\right) - U_{0,0}\right]$$

が成り立つことになる。

$j = 0, 1, \cdots, \frac{N_\theta}{4} - 1$ について平均をとる。

$$\triangle u(0, 0, n\tau) = \frac{4}{h_r^2} \left(\frac{1}{N_\theta} \sum_{j=0}^{N_\theta - 1} U_{1,j} - U_{0,0} \right)$$

また、

$$\frac{\partial u}{\partial t}(0, 0, n\tau) \fallingdotseq \frac{U_{0,0}^{n+1} - U_{0,0}^n}{\tau}.$$

よって

$$U_{0,0}^{n+1} = (1 - 4\lambda_r) U_{0,0}^n + \frac{4\lambda_r}{N_\theta} \sum_{j=0}^{N_\theta - 1} U_{1,j}^n \qquad (j = 0, 1, \cdots, N_\theta - 1; n = 0, 1, 2, \cdots)$$

と差分方程式が導かれる。

2.1.3 陽解法の安定性

この円盤の熱伝導方程式を陽解法で解くにあたり、遠藤・高木・内藤 [6] では、安定条件は

$$\min_{1 \le i \le N_r - 1} \left(1 - 2\lambda_r - \frac{\lambda_\theta}{r_i^2} \right) \ge 0$$

ではないかと予想した。(長方形領域の場合との類推で、(2.1) の係数がすべて 0 以上になることから導いた)。これは

$$0 \le 1 - 2\lambda_r - \frac{\lambda_\theta}{r_1^2} = 1 - \frac{2\tau}{h_r^2} - \frac{\tau}{h_\theta^2 h_r^2} = 1 - \tau \left(\frac{2}{h_r^2} + \frac{1}{h_\theta^2 h_r^2} \right)$$

すなわち

$$\tau \le \left(\frac{2}{h_r^2} + \frac{1}{h_\theta^2 h_r^2} \right)^{-1} = \frac{h_\theta^2 h_r^2}{2h_\theta^2 + 1}$$

となる。

2.1.4 実験結果

次に、実際に数値実験した結果を以下に記す。
この問題における厳密解は

$$u(r, \theta, t) = \sum_{n=0}^{\infty} \sum_{m=1}^{\infty} e^{-\mu_{m,n} t} J_n(\mu_{m,n} r)(A_{nm} \cos n\theta + B_{nm} \sin n\theta)$$

という式で求めることが出来る。また A_{nm}, B_{nm} は係数である。（詳細は遠藤・高木・内藤 [6] を見よ）

まず、厳密解を $u(r,\theta,t) = J_0(\mu_{10}r)e^{-\mu_{10}t}$ とする。$N_r = 20, N_\theta = 80$ の場合、予想安定条件は $\tau \leq 7.66336\times10^{-6}$ であるが、実際に τ を色々変えて試すと以下のようになった。

τ	安定 *or* 不安定
1.035e-03	不安定
1.034e-03	不安定
1.0339e-03	安定
1.03e-03	安定
1.0e-03	安定
7.66336e-06	安定

分割数を変えて、$N_r = 20, N_\theta = 40$ の場合、予想安定条件は $\tau \leq 3.00998$e-05

τ	安定 *or* 不安定
1.0339e-03	不安定
1.03387e-03	不安定
1.03385e-03	安定
1.03383e-03	安定
1.033e-03	安定
1e-03	安定
3.00998e-05	安定

次に、厳密解を $u(r,\theta,t) = e^{-\mu_{10}t}J_0(\mu_{1,0}r) + e^{-\mu_{11}t}J_1(\mu_{1,1}r)(\cos\theta + \sin\theta)$ とする。$N_r = 20, N_\theta = 80$ の場合、予想の安定条件は $\tau \leq$ 7.66336e-06 である。

τ	安定 *or* 不安定
7.69e-06	不安定
7.689e-06	不安定
7.688e-06	不安定
7.687e-06	安定
7.685e-06	安定
7.68e-06	安定
7.66336e-06	安定

$N_r = 20, N_\theta = 40$ の場合、予想安定条件は $\tau \leq$ 3.00998e-05 である。

τ	安定 *or* 不安定
3.048e-05	不安定
3.0479e-05	不安定
3.0478e-05	安定
3.0477e-05	安定
3.047e-05	安定
3.03e-05	安定
3.0998e-05	安定

厳密解を $u(r,\theta,t) = e^{-\mu_{1,0}t}J_0(\mu_{1,0}r)$ とした時は、予想した安定条件よりも 10^2 も大きい τ をとっても安定という結果が得られた。一方、厳密解を $u(r,\theta,t) = e^{-\mu_{1,0}t}J_0(\mu_{1,0}r) + e^{-\mu_{1,1}t}J_1(\mu_{1,1}r)(\cos\theta + \sin\theta)$ とした時は、予想の τ に近い結果になった。また、$h \overset{\text{def.}}{=} \max\{h_r, h_\theta\}$ とおくと、$\tau = O(h^4)$ となる。長方形領域の場合と比べても、τ を非常に小さくとらなければいけない。計算もかなり時間がかかってしまうこともあり、この条件は非常に厳しいと言える。よって、円盤領域における熱伝導方程式を数値解析するために陽解法を用いるのは解の安定性から考えてあまり進められないようである。

2.1.5 陽解法のプログラム

```
/*
 * heat2d-disk-e.c --- 円盤における熱方程式を陽解法で解く
 *
 *      To compile this program on WS's in 6701,
 *              gcc -I/usr/local/include -c call_gnuplot.c
 *              ccmg heat2d-disk-e.c call_gnuplot.o
 *      Sample input

oyabun% ./heat2d-e2
Nr, Nt: 20 80
Tmax: 1
τ (≦ 7.66336e-06): 5e-6
λ=0.00281057 になりました。
Δ t(≧ 5e-06): 5e-3

 */

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

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

#include "call_gnuplot.h"

double u0(double, double);
double exactu(double, double, double);
double maxnorm(int, int, matrix);

int main()
{
  int Nr, Nt, i, j, n, skip, nMax;
  double pi, hr, ht, ri, ri2, theta_j, M, ex;
  double lambda_r, lambda_t, lambda;
  double tau, Tmax, t, dt, s;
  matrix u, newu;
  char label[100];
  /* 次の変数を 0 にすると原点だけで誤差を計算する。
   * 0 以外の値（例えば 1）にすると円盤全体で誤差を計算する。
   */
  int zentai = 1;

  pi = 4.0 * atan(1.0);
  /* 分割数を決定 */
  printf("Nr, Nt: ");
  scanf("%d %d", &Nr, &Nt);
  if ((u = new_matrix(Nr + 1, Nt + 1)) == NULL) {
```

```
  fprintf(stderr, "配列 u を確保できませんでした。");
  exit(1);
}
if ((newu = new_matrix(Nr + 1, Nt + 1)) == NULL) {
  fprintf(stderr, "配列 newu を確保できませんでした。");
  exit(1);
}
/* 最終時刻の決定 */
printf("Tmax: ");
scanf("%lf", &Tmax);

/* 刻み幅 */
hr = 1.0 / Nr;
ht = 2 * pi / Nt;

/* 時間刻み幅の決定 */
printf("τ (≦%g): ",
       0.5 * (hr * hr * ht * ht) / (1 + ht * ht));
scanf("%lf", &tau);

/* λ r, λ θ */
lambda_r = tau / (hr * hr);
lambda_t = tau / (ht * ht);
lambda = lambda_r + lambda_t;

printf("λ=%g になりました。\n", lambda);

/* 結果を出力する時間間隔を決定 */
printf("Δ t(≧%g): ", tau);
scanf("%lf", &dt);
if (dt < tau) {
  dt = tau;
  printf("Δ=%g\n", dt);
}
skip = rint(dt / tau);

/* GNUPLOT の準備 */
open_gnuplot();

/* 初期値の設定 */
for (i = 0; i <= Nr; i++) {
  ri = i * hr;
  for (j = 0; j <= Nt; j++) {
    theta_j = j * ht;
    u[i][j] = u0(ri, theta_j);
  }
}

/* ループ */
```

```
nMax = rint(Tmax / tau);
for (n = 1; n <= nMax; n++) {
  for (i = 1; i < Nr; i++) {
    ri = i * hr;
    ri2 = ri * ri;
    for (j = 0; j < Nt; j++) {
      int jm1 = j - 1;
      if (jm1 == -1)
        jm1 = Nt - 1;
      newu[i][j] = (1 - 2 * lambda_r - 2 * lambda_t / (ri2)) * u[i][j]
        + lambda_r * (u[i + 1][j] + u[i - 1][j])
        + lambda_r * hr / (2 * ri) * (u[i + 1][j] - u[i - 1][j])
        + lambda_t / (ri2) * (u[i][j + 1] + u[i][jm1]);
    }
  }
  for (i = 1; i < Nr; i++)
    newu[i][Nt] = newu[i][0];
  /* Dirichlet 境界条件 */
  for (j = 0; j <= Nt; j++)
    newu[Nr][j] = 0.0;

  /* 原点での値 */
  s = 0;
  for (j = 0; j < Nt; j++)
    s += u[1][j];

  newu[0][0] = lambda_r * (4 * (s / Nt - u[0][0])) + u[0][0];
  for (j = 1; j <= Nt; j++)
    newu[0][j] = newu[0][0];

  /* 値の更新 */
  for (i = 0; i <= Nr; i++)
    for (j = 0; j <= Nt; j++)
      u[i][j] = newu[i][j];

  t = tau * n;

  /* 誤差を測る */
  if (zentai) {
    M = 0.0;
    for (i = 0; i <= Nr; i++) {
      ri = i * hr;
      for (j = 0; j <= Nt; j++) {
        double e;
        theta_j = j * ht;
        e = fabs(exactu(ri, theta_j, t) - u[i][j]);
        if (e > M)
          M = e;
      }
```

```
      }
      printf("n=%d, norm=%g, t=%g, 誤差=%g\n",
             n, maxnorm(Nr + 1, Nt + 1, u), t, M);
    }
    else {
      ex = exactu(0.0, 0.0, t);
      M = fabs(u[0][0] - ex);
      printf("n=%d, norm=%g, t=%g, exactu=%g , 誤差=%g\n",
             n, maxnorm(Nr + 1, Nt + 1, u), t, ex, M);
    }

    /* Δ t の整数倍の時刻ではグラフを描く */
    if (n % skip == 0) {
      t = tau * n;
      sprintf(label, "t=%g", t);
      disk(Nr, Nt, u, label);
    }
  }

  close_gnuplot();

  return 0;
}

/* 厳密解を計算する関数 */

#define mu01    (2.404825557695773)

double exactu(double r, double theta, double t)
{
  return exp(- mu01 * mu01 * t) * j0(mu01 * r);
}

/* 初期データ */
double u0(double r, double theta)
{
  return j0(mu01 * r);
}

double maxnorm(int m, int n, matrix u)
{
  int i, j, i0, j0;
  double tmpmax, absu;
  i0 = 0;
  j0 = 0;
  tmpmax = fabs(u[0][0]);
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      if ((absu = fabs(u[i][j])) > tmpmax) {
```

```
        tmpmax = absu;
        i0 = i;
        j0 = j;
      }
  printf("(i,j)=(%d,%d)", i0, j0);
  return tmpmax;
}
```

2.2 半陰スキーム

2.2.1 差分方程式の導出

先程の陽解法において、安定性の条件を厳しくしているのは

$$\triangle = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2}$$

の式の $\frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2}$ という項が原因である。そこで、この項を左辺に持ってくるために θ に関する導関数の部分のみを陰的に扱うことにする。その他は陽解法の時と同様にして

$$\frac{U_{i,j}^{n+1} - U_{i,j}^{n}}{\tau} = \frac{U_{i+1,j}^{n} - 2U_{i,j}^{n} + U_{i-1,j}^{n}}{h_r^2} + \frac{1}{r_i}\frac{U_{i+1,j}^{n} - U_{i-1,j}^{n}}{2h_r} + \frac{1}{r_i^2}\frac{U_{i,j+1}^{n+1} - 2U^{n+1}i, j + U_{i,j-1}^{n+1}}{h_\theta^2}$$

が導かれる。分母を払って

$$U_{i,j}^{n+1} - U_{i,j}^{n} = \lambda_r(U_{i+1,j}^{n} - 2U_{i,j}^{n} + U_{i-1,j}^{n}) + \frac{\lambda_r h_r}{2r_i}(U_{i+1,j}^{n} - U_{i-1,j}^{n}) + \frac{\lambda_\theta}{r_i^2}(U_{i,j+1}^{n+1} - 2U_{i,j}^{n+1} + U_{i,j-1}^{n+1})$$

これより

$$\begin{aligned}
&\left(1 + \frac{2\lambda_\theta}{r_i^2}\right) U_{i,j}^{n+1} - \frac{\lambda_\theta}{r_i^2}\left(U_{i,j+1}^{n+1} + U_{i,j-1}^{n+1}\right) \\
&\quad = (1 - 2\lambda_r)U_{i,j}^{n} + \lambda_r\left[\left(1 + \frac{h_r}{2r_i}\right) U_{i+1,j}^{n} + \left(1 - \frac{h_r}{2r_i}\right) U_{i-1,j}^{n}\right] \\
&\quad (i = 1, 2, \cdots, N_r - 1; j = 0, 1, \cdots, N_\theta - 1; n = 0, 1, 2, \cdots)
\end{aligned}$$

という差分方程式を得る。

ただし、ここでも $U_{i,-1}^{n+1} = U_{i,N_\theta - 1}^{n+1}$、$U_{i,N_\theta}^{n+1} = U_{i,0}^{n+1}$ と考える。

次に、N_θ 次の正方行列 $A_i (i = 1, 2, \cdots, N_r - 1)$ を

$$A_i \overset{\text{def.}}{=} \left(1 + \frac{2\lambda_\theta}{r_i^2}\right) I - \frac{\lambda_\theta}{r_i^2} J$$

と定める。ただし、I は N_θ 次の単位行列で

$$J \overset{\text{def.}}{=} \begin{pmatrix} 0 & 1 & & & & & 1 \\ 1 & 0 & 1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & 1 & 0 & 1 & \\ 1 & & & & 1 & 0 & \end{pmatrix}$$

の形の N_θ 次正方行列とする。また

$$U_i^{n+1} \overset{\text{def.}}{=} \begin{pmatrix} U_{i,0}^{n+1} \\ U_{i,1}^{n+1} \\ \vdots \\ U_{i,N_\theta-1}^{n+1} \end{pmatrix},$$

$$b_i^n = \begin{pmatrix} (1-2\lambda_r)U_{i,0}^n + \lambda_r \left[\left(1+\frac{h_r}{2r_i}\right) U_{i+1,0}^n + \left(1-\frac{h_r}{2r_i}\right) U_{i-1,0}^n \right] \\ \vdots \\ (1-2\lambda_r)U_{i,j}^n + \lambda_r \left[\left(1+\frac{h_r}{2r_i}\right) U_{i+1,j}^n + \left(1-\frac{h_r}{2r_i}\right) U_{i-1,j}^n \right] \\ \vdots \\ (1-2\lambda_r)U_{i,N_\theta-1}^n + \lambda_r \left[\left(1+\frac{h_r}{2r_i}\right) U_{i+1,N_\theta-1}^n + \left(1-\frac{h_r}{2r_i}\right) U_{i-1,N_\theta-1}^n \right] \end{pmatrix}$$

とおくと

$$A_i U_i^{n+1} = b_i^n \qquad (i = 1, 2, \cdots, N_r - 1)$$

と書ける。

原点においては、陽解法と同様に

$$U_{0,0}^{n+1} = (1-4\lambda_r)U_{0,0}^n + \frac{4\lambda_r}{N_\theta} \sum_{j=0}^{N_\theta-1} U_{1,j}^n.$$

また、境界条件より

$$U_{N_r,j}^{n+1} = f(\cos\theta_j, \sin\theta_j) \qquad (j = 0, 1, \cdots, N_\theta - 1)$$

2.2.2 半陰スキームの安定性

陽解法の場合と同じように類推すると、このスキームが安定であるためには

$$1 - 2\lambda_r \geq 0$$

が必要ではないかと推測される。つまり

$$\tau \leq \frac{h_r^2}{2}.$$

2.2.3 実験結果

ここで実際に数値実験した結果を以下に記す。

厳密解を $u(r,\theta,t) = e^{-\mu_{1,0}t}J_0(\mu_{1,0}r)$ とする。$N_r = 20, N_\theta = 80$ の場合、予想安定条件は $\tau \leq 0.00125$ であるが、実際に τ を色々変えて試すと以下のようになった。

τ	安定 *or* 不安定
0.00104	不安定
0.001037	不安定
0.001036	不安定
0.001035	安定
0.001033	安定
0.001	安定
0.0008	安定

次に、$N_r = 20, N_\theta = 40$ の場合、予想安定条件は $\tau \leq 0.00125$ である。

τ	安定 *or* 不安定
0.00104	不安定
0.001037	不安定
0.001036	不安定
0.001035	安定
0.001033	安定
0.001	安定
0.0008	安定

厳密解を $u(r,\theta,t) = e^{-\mu_{1,0}t}J_0(\mu_{1,0}r) + e^{-\mu_{1,1}t}J_1(\mu_{1,1}r)(\cos\theta + \sin\theta)$ とする。

$N_r = 20, N_\theta = 80$ の場合、予想安定条件は $\tau \leq 0.00125$ である。

τ	安定 *or* 不安定
0.00104	不安定
0.001037	不安定
0.001036	不安定
0.001035	安定
0.001033	安定
0.001	安定
0.0008	安定

次に $N_r = 20, N_\theta = 40$ の場合、予想安定条件は $\tau \leq 0.00125$ である。

τ	安定 *or* 不安定
0.00104	不安定
0.001037	不安定
0.001036	不安定
0.001035	安定
0.001033	安定
0.001	安定
0.0008	安定

先程の陽解法の場合と比べて、半陰スキームでは τ をかなり大きくしても安定になった。また予想の条件に近い結果が得られた。初期条件を変化させても安定条件は変わらなかった。

2.2.4 半陰スキームのプログラム

```
/*半陰スキームで熱方程式を解く
 * heat2d-disk-i.c -- solve the heat equation in a two dimensional disk
 * by semi-implicit finite difference method.
 *
 *      To compile this program on WS's in 6701,
 *              gcc -I/usr/local/include -c call_gnuplot.c call_ptrilu.c
 *      or try
 *              ccmg heat2d-disk-i.c ptrilu.o
 */
```

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

/* to use matrix, new_matrix() */
#include <matrix.h>
/* to use gnuplot */
#include "call_gnuplot.h"
/* to use ptrilu() */
#include "ptrilu.h"

double u0(double, double);
double maxnorm(int, int, matrix);
double exactu(double, double, double);

double pi;

int main()
{
  int Nr, Nt, i, j, n, skip, nMax;
  double hr, ht, ri, theta_j;
  double lambda_r, lambda_t;
  double tau, tau_limit, tau_limit2, Tmax, t, dt, s, ex, M;
  matrix u, newu;
  vector *al, *ad, *au, *ab, *ar, b;
  char label[100];
  /* 次の変数を 0 にすると原点だけで誤差を計算する。
   * 0 以外の値（例えば 1）にすると円盤全体で誤差を計算する。
   */
  int zentai = 1;

  pi = 4.0 * atan(1.0);

  /* 分割の細かさを指定する */
  printf("Nr, Nt: ");
  scanf("%d %d", &Nr, &Nt);
  /* 差分解を記憶する変数の準備 */
  if ((u = new_matrix(Nr + 1, Nt + 1)) == NULL) {
    fprintf(stderr, "配列 u を確保できませんでした。");
    exit(1);
  }
  if ((newu = new_matrix(Nr + 1, Nt + 1)) == NULL) {
    fprintf(stderr, "配列 newu を確保できませんでした。");
    exit(1);
  }
  /* 連立 1 次方程式の係数行列を記憶するための変数の準備 */
  al = (vector *) malloc(sizeof(vector *) * Nr);
  ad = (vector *) malloc(sizeof(vector *) * Nr);
  au = (vector *) malloc(sizeof(vector *) * Nr);
  ab = (vector *) malloc(sizeof(vector *) * Nr);
```

```
ar = (vector *) malloc(sizeof(vector *) * Nr);
for (i = 1; i < Nr; i++) {
  if ((al[i] = new_vector(Nt)) == NULL)
    perror("cannot allocate matirx.");
  if ((ad[i] = new_vector(Nt)) == NULL)
    perror("cannot allocate matirx.");
  if ((au[i] = new_vector(Nt)) == NULL)
    perror("cannot allocate matirx.");
  if ((ab[i] = new_vector(Nt)) == NULL)
    perror("cannot allocate matirx.");
  if ((ar[i] = new_vector(Nt)) == NULL)
    perror("cannot allocate matirx.");
}
if ((b = new_vector(Nt + 1)) == NULL)
  perror("cannot allocate matirx.");

printf("Tmax: ");
scanf("%lf", &Tmax);

hr = 1.0 / Nr;
ht = 2 * pi / Nt;

tau_limit = 0.5 * (hr * hr * ht * ht) / (1 + ht * ht);
tau_limit2 = 0.25 * hr * hr;
printf("τ (陽解法の場合の上限=%g, λ r ≦ 1/4 のための上限=%g): ",
       tau_limit, tau_limit2);
scanf("%lf", &tau);

lambda_r = tau / (hr * hr);
lambda_t = tau / (ht * ht);

printf("Δ t: ");
scanf("%lf", &dt);
if (dt < tau) {
  dt = tau;
  printf("Δ t=%g\n", dt);
}
skip = rint(dt / tau);

open_gnuplot();

for (i = 0; i <= Nr; i++) {
  ri = i * hr;
  for (j = 0; j <= Nt; j++) {
    theta_j = j * ht;
    u[i][j] = u0(ri, theta_j);
  }
}
/* 係数行列に値をセットして、LU 分解しておく */
```

```
for (i = 1; i < Nr; i++) {
  double ri, ri2, d, od;
  ri = i * hr;
  ri2 = ri * ri;
  d = 1 + 2 * lambda_t / ri2;
  od = -lambda_t / ri2;
  for (j = 0; j < Nt; j++) {
    ad[i][j] = d;
    al[i][j] = au[i][j] = od;
    ab[i][j] = ar[i][j] = 0.0;
  }
  ab[i][0] = ab[i][Nt - 2] = ar[i][0] = ar[i][Nt - 2] = od;
  ptrilu0(Nt, al[i], ad[i], au[i], ab[i], ar[i]);
}

/* 時間に関するループ */
nMax = rint(Tmax / tau);
for (n = 1; n <= nMax; n++) {
  for (i = 1; i < Nr; i++) {
    double alpha, beta;
    /* 右辺を作る */
    alpha = lambda_r * (1.0 + 0.5 / i);
    beta = lambda_r * (1.0 - 0.5 / i);
    for (j = 0; j < Nt; j++) {
      b[j] = (1 - 2 * lambda_r) * u[i][j] +
        alpha * u[i + 1][j] + beta * u[i - 1][j];
    }
    /* 連立 1 次方程式を解く */
    ptrisol0(Nt, al[i], ad[i], au[i], ab[i], ar[i], b);
    /* コピーする */
    for (j = 0; j < Nt; j++)
      newu[i][j] = b[j];
    newu[i][Nt] = b[0];
  }
  /* 同次 Dirichlet 境界条件 */
  for (j = 0; j <= Nt; j++)
    newu[Nr][j] = 0.0;

  /* 原点 */
  s = 0.0;
  for (j = 0; j < Nt; j++)
    s += u[1][j];

  newu[0][0] = 4 * lambda_r * (s / Nt - u[0][0]) + u[0][0];
  for (j = 1; j <= Nt; j++)
    newu[0][j] = newu[0][0];

  /* 更新 */
  for (i = 0; i <= Nr; i++)
```

```
      for (j = 0; j <= Nt; j++)
        u[i][j] = newu[i][j];

    t = n * tau;
    /* 誤差を測る */
    if (zentai) {
      M = 0.0;
      for (i = 0; i <= Nr; i++) {
        ri = i * hr;
        for (j = 0; j <= Nt; j++) {
          double e;
          theta_j = j * ht;
          e = fabs(exactu(ri, theta_j, t) - u[i][j]);
          if (e > M)
            M = e;
        }
      }
      printf("n=%d, norm=%g, t=%g, 誤差=%g\n",
             n, maxnorm(Nr + 1, Nt + 1, u), t, M);
    }
    else {
      ex = exactu(0.0, 0.0, t);
      M = fabs(u[0][0] - ex);
      printf("n=%d, norm=%g, t=%g, exactu=%g , 誤差=%g\n",
             n, maxnorm(Nr + 1, Nt + 1, u), t, ex, M);
    }

    if (n % skip == 0) {
      t = n * tau;
      sprintf(label, "t=%g", t);
      disk(Nr, Nt, u, label);
    }
  }

  close_gnuplot();

  return 0;
}

#define mu01 (2.404825557695773)

/* 初期値 */
double u0(double r, double theta)
{
  return j0(mu01 * r);
}

/* 厳密解 */
double exactu(double r, double theta, double t)
```

```
{
  return exp(- mu01 * mu01 * t) * j0(mu01 * r);
}

double maxnorm(int m, int n, matrix u)
{
  int i, j, i0, j0;
  double tmpmax, absu;
  i0 = 0;
  j0 = 0;
  tmpmax = fabs(u[0][0]);
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      if ((absu = fabs(u[i][j])) > tmpmax) {
        tmpmax = absu;
        i0 = i;
        j0 = j;
      }
  printf("(i,j)=(%d,%d)", i0, j0);
  return tmpmax;
}
```

2.3 ADI法

ADI 法とは、時間に関する 1 ステップを半分に分割し、各半ステップ毎交互に陰解法を適用する方向を入れ換えるという方法である。

2.3.1 差分方程式の導出

第 n 段から第 $n+\frac{1}{2}$ 段

まず θ に関する導関数の部分を陰的に扱う。すると

$$\begin{aligned}\frac{U_{i,j}^{n+\frac{1}{2}}-U_{i,j}^{n}}{\tau/2} &= \frac{U_{i+1,j}^{n}-2U_{i,j}^{n}+U_{i-1,j}^{n}}{h_r^2} \\ &\quad +\frac{1}{r_i}\frac{U_{i+1,j}^{n}-U_{i-1,j}^{n}}{2h_r}+\frac{1}{r_i^2}\frac{U_{i,j+1}^{n+\frac{1}{2}}-2U_{i,j}^{n+\frac{1}{2}}+U_{i,j+1}^{n+\frac{1}{2}}}{h_\theta^2}\end{aligned}$$

$\lambda_r'=\tau/2h_r^2=\lambda_r/2,\quad \lambda_\theta'=\tau/2h_\theta^2=\lambda_\theta/2$ とおくと

$$\begin{aligned}U_{i,j}^{n+\frac{1}{2}}-U_{i,j}^{n} &= \lambda_r'(U_{i+1,j}^{n}-2U_{i,j}^{n}+U_{i-1,j}^{n}) \\ &\quad +\lambda_r'\frac{h_r}{2r_i}(U_{i+1,j}^{n}-U_{i-1,j}^{n})+\frac{\lambda_\theta'}{r_i^2}(U_{i,j+1}^{n+\frac{1}{2}}-2U_{i,j}^{n+\frac{1}{2}}+U_{i,j-1}^{n+\frac{1}{2}})\end{aligned}$$

これは半陰スキームと同じ形で

$$\left(1+\frac{2\lambda'_\theta}{r_i^2}\right)U_{i,j}^{n+\frac{1}{2}}-\frac{\lambda'_\theta}{r_i^2}(U_{i,j+1}^{n+\frac{1}{2}}+U_{i,j-1}^{n+\frac{1}{2}})$$
$$=(1-2\lambda'_r)U_{i,j}^n+\lambda'_r\left[\left(1+\frac{h_r}{2r_i}\right)U_{i+1,j}^n+\left(1-\frac{h_r}{2r_i}\right)U_{i-1,j}^n\right]$$
$$(i=1,2,\cdots,N_r-1;j=0,1,\cdots,N_\theta-1;n=0,1,\cdots)$$

となる。ただし、$U_{i,-1}^{n+\frac{1}{2}}=U_{i,N_\theta-1}^{n+\frac{1}{2}},U_{i,N_\theta}^{n+\frac{1}{2}}=U_{i,0}^{n+\frac{1}{2}}$ とする。

次に

$$A'_i=\left(1+\frac{2\lambda'_\theta}{r_i^2}\right)I-\frac{\lambda'_\theta}{r_i^2}J$$

とおく。ただし、I,J は半陰スキームでの行列と同じである。

また

$$U_i^{n+\frac{1}{2}}=\begin{pmatrix}U_{i,0}^{n+\frac{1}{2}}\\U_{i,1}^{n+\frac{1}{2}}\\\vdots\\U_{i,N_\theta-1}^{n+\frac{1}{2}}\end{pmatrix},b_i^n=\begin{pmatrix}(1-2\lambda'_r)U_{i,0}^n+\lambda'_r\left[\left(1+\frac{h_r}{2r_i}\right)U_{i+1,0}^n+\left(1-\frac{h_r}{2r_i}\right)U_{i-1,0}^n\right]\\\vdots\\(1-2\lambda'_r)U_{i,j}^n+\lambda'_r\left[\left(1+\frac{h_r}{2r_i}\right)U_{i+1,j}^n+\left(1-\frac{h_r}{2r_i}\right)U_{i-1,j}^n\right]\\\vdots\\(1-2\lambda'_r)U_{i,N_\theta-1}^n+\lambda'_r\left[\left(1+\frac{h_r}{2r_i}\right)U_{i+1,N_\theta-1}^n+\left(1-\frac{h_r}{2r_i}\right)U_{i-1,N_\theta-1}^n\right]\end{pmatrix}$$

とおくと

$$A_iU_i^{n+\frac{1}{2}}=b_i^n\qquad(i=1,2,\cdots,N_r-1)$$

と書ける。また原点においては、陽解法のときと同様に

$$U_{0,0}^{n+\frac{1}{2}}=(1-4\lambda'_r)U_{0,0}^n+\frac{4\lambda'_r}{N_\theta}\sum_{j=0}^{N_\theta-1}U_{1,j}^n\qquad(j=0,1,\cdots,N_\theta-1;n=0,1,\cdots)$$

とする。

第$n+1/2$段から第$n+1$段今度はrに関する導関数の部分を陰的に扱う。

$$\frac{U_{i,j}^{n+1}-U_{i,j}^{n+\frac{1}{2}}}{\tau/2}=\frac{U_{i+1,j}^{n+1}-2U_{i,j}^{n+1}+U_{i-1,j}^{n+1}}{h_r^2}$$
$$+\frac{1}{r_i}\frac{U_{i+1,j}^{n+1}-U_{i-1,j}^{n+1}}{2h_r}+\frac{1}{r_i^2}\frac{U_{i,j+1}^{n+\frac{1}{2}}-2U_{i,j}^{n+\frac{1}{2}}+U_{i,j-1}^{n+\frac{1}{2}}}{h_\theta^2}$$

分母を払って

$$\begin{aligned} U_{i,j}^{n+1} - U_{i,j}^{n+\frac{1}{2}} &= \lambda_r'(U_{i+1,j}^{n+1} - 2U_{i,j}^{n+1} + U_{i-1,j}^{n+1}) \\ &\quad +\lambda_r' \frac{h_r}{2r_i}(U_{i+1,j}^{n+1} - U_{i-1,j}^{n+1}) + \frac{\lambda_\theta'}{r_i^2}(U_{i,j+1}^{n+\frac{1}{2}} - 2U_{i,j}^{n+\frac{1}{2}} + U_{i,j-1}^{n+\frac{1}{2}}) \end{aligned}$$

移項して整理すると

$$\begin{aligned} (1+2\lambda_r')U_{i,j}^{n+1} - \lambda_r' \left[\left(1 + \frac{h_r}{2r_i}\right) U_{i+1,j}^{n+1} + \left(1 - \frac{h_r}{2r_i}\right) U_{i-1,j}^{n+1} \right] \\ = \left(1 - \frac{2\lambda_\theta'}{r_i^2}\right) U_{i,j}^{n+\frac{1}{2}} + \frac{\lambda_\theta'}{r_i^2}(U_{i,j+1}^{n+\frac{1}{2}} + U_{i,j-1}^{n+\frac{1}{2}}) \end{aligned}$$

ここで、左辺で $i=1$ のとき $U_{0,j}^{n+1}$（原点での値）、$i=N_r-1$ のとき $U_{N_r,j}^{n+1}$（境界 $r=1$ での値）が現れるが、これは右辺に移項する。

$$B = \begin{pmatrix} 1+2\lambda_r' & -\lambda_r'(1+\frac{h_r}{2r_1}) & & & \\ -\lambda_r'(1-\frac{h_r}{2r_2}) & 1+2\lambda_r' & -\lambda_r'(1+\frac{h_r}{2r_2}) & & \\ & \ddots & \ddots & \ddots & \\ & & -\lambda_r'(1-\frac{h_r}{2r_{N_r-2}}) & 1+2\lambda_r' & -\lambda_r'(1+\frac{h_r}{2r_{N_r-2}}) \\ & & & -\lambda_r'(1-\frac{h_r}{2r_{N_r-1}}) & 1+2\lambda_r' \end{pmatrix},$$

$$U_j^{n+1} = \left(U_{1,j}^{n+1}, U_{2,j}^{n+1}, \cdots, U_{N_r-1,j}^{n+1}\right)^T,$$

$$c_j^{n+\frac{1}{2}} = \begin{pmatrix} \left(1 - \frac{2\lambda_\theta'}{r_1^2}\right) U_{1,j}^{n+\frac{1}{2}} + \frac{\lambda_\theta'}{r_1^2}(U_{1,j+1}^{n+\frac{1}{2}} + U_{1,j-1}^{n+\frac{1}{2}}) \\ \vdots \\ \left(1 - \frac{2\lambda_\theta'}{r_i^2}\right) U_{i,j}^{n+\frac{1}{2}} + \frac{\lambda_\theta'}{r_i^2}(U_{i,j+1}^{n+\frac{1}{2}} + U_{i,j-1}^{n+\frac{1}{2}}) \\ \vdots \\ \left(1 - \frac{2\lambda_\theta'}{r_{N_r-1}^2}\right) U_{N_r-1,j}^{n+\frac{1}{2}} + \frac{\lambda_\theta'}{r_{N_r-1}^2}(U_{N_r-1,j+1}^{n+\frac{1}{2}} + U_{N_r-1,j-1}^{n+\frac{1}{2}}) \end{pmatrix} + \begin{pmatrix} \lambda_r' \left(1 - \frac{h_r}{2r_1}\right) U_{0,j}^{n+1} \\ 0 \\ \vdots \\ 0 \\ \lambda_r' \left(1 - \frac{h_r}{2r_{N_r-1}}\right) U_{N_r,j}^{n+1} \end{pmatrix}$$

とおくと

$$BU_j^{n+1} = c_j^{n+\frac{1}{2}} \qquad (j = 0, 1, \cdots, N_\theta - 1)$$

と書ける。$U_{0,j}^{n+1}$ は原点での値であるが、これは陽解法で

$$U_{0,j}^{n+1} = U_{0,0}^{n+1} = (1-4\lambda_r')U_{0,0}^{n+\frac{1}{2}} + \frac{4\lambda_r'}{N_\theta} \sum_{j=0}^{N_\theta-1} U_{1,j}^{n+\frac{1}{2}}$$

と与える。$U_{N_r,j}^{n+1}$ は境界上の値であるから

$$U_{N_r,j}^{n+1} = f(\cos\theta_j, \sin\theta_j) \qquad (j = 0, 1, \cdots, N_\theta - 1)$$

とする。

2.3.2 ADI 法の安定性

長方形領域では、ADI 法にすると無条件に安定になることはよく知られている。円盤でも同様に無条件安定になると期待されたが、次に紹介する数値実験の結果ではこれが翻された。

結果を見ると

$$\tau \leq h_r^2$$

が安定条件になっているように見える。

2.3.3 実験結果

厳密解を $u(r,\theta,t) = e^{-\mu_{1,0}t} J_0(\mu_{1,0} r)$ とする。
$N_r = 20, N_\theta = 80$ の場合、予想安定条件は $\tau \leq 0.0025$ であるが、実際に τ を色々試すと以下のようになった。

τ	安定 *or* 不安定
0.0025	不安定
0.002495	不安定
0.002493	不安定
0.002492	安定
0.00249	安定
0.001	安定

$N_r = 40, N_\theta$ の場合、予想安定条件は $\tau \leq 0.000625$ である。

τ	安定 *or* 不安定
0.00063	不安定
0.000627	不安定
0.000625	不安定
0.000624	安定
0.00062	安定
0.0006	安定

ADI 法では、これまで紹介した差分法の中で最も τ の値を大きくとることが出来た。さらに予想した安定条件に非常に近い結果も得られた。

2.3.4 ADI 法のプログラム

```
/*
 * heat2d-disk-adi.c --- ADI 法により二次元円盤の熱方程式を解くプログラム
 *
 *  参考: http://www.math.meiji.ac.jp/~mk/labo/text/heat-fdm-2.pdf
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "ptrilu.h"
#include "trid-lu.h"
/* to use gnuplot */
#define USEGNUPLOT
#ifdef USEGNUPLOT
#include "call_gnuplot.h"
#endif

#define BESSEL

/* 初期値 */
double u0(double, double);
/* 境界値 */
double boundary_data(double, double, double);
/* 厳密解 */
double exactu(double r, double theta, double t);
/* ノルム計算 */
double maxnorm(int m, int n, matrix u);
/* 円周率 */
double PI;

int main()
{
  int n, i, j, nMax, NN, Nr, Nt;
  double tau, t, Tmax, hr, ht, dt;
  double ri, rr, coef_a, coef_b, coef_c, lambda_r, lambda_t, s;
  vector b, Bd, Bl, Bu;
  matrix Ad, Al, Au, Ab, Ar, U, UU;
  int zentai = 1, output = 0;
  double MaxError;
  double tau_limit,tau_limit2;
  int skip;
  char label[100];

  PI = 4.0 * atan(1.0);

  /* いつまで計算するか */
```

```
 printf("Tmax: "); scanf("%lf", &Tmax);

 /* r, θ 各方向の分割数 */
 printf("Nr, N θ : ");
 scanf("%d%d", &Nr, &Nt);

 /* 刻み幅 */
 hr = 1.0 / Nr;
 ht = 2 * PI / Nt;

 tau_limit = 0.5 * (hr * hr * ht * ht) / (1 + ht * ht);
 tau_limit2 = 0.25 * hr * hr;
 printf("τ (陽解法の場合の上限=%g, λ r ≦ 1/4 のための上限=%g): ",
        tau_limit, tau_limit2);
 printf("τ : "); scanf("%lf", &tau);

 printf("結果を出力する時間間隔: "); scanf("%lf", &dt);
 skip = rint(dt / tau);

 /* λ r, λ θ */
 lambda_r = tau / (2 * hr * hr);
 lambda_t = tau / (2 * ht * ht);

 /* メモリの確保 */

 /* 連立 1 次方程式を解くための作業ベクトル */
 NN = (Nt > Nr) ? Nt : Nr;
 b = new_vector(NN + 1);
 /* 差分解 */
 U = new_matrix(Nr + 1, Nt + 1);
/* 差分解 (作業用) */
 UU = new_matrix(Nr + 1, Nt + 1);
 /* 連立方程式の係数行列 B */
 Bl = new_vector(Nr + 1);
 Bd = new_vector(Nr + 1);
 Bu = new_vector(Nr + 1);
 /* 連立方程式の行列 Ai */
 Al = new_matrix(Nr + 1, Nt+1);
 Ad = new_matrix(Nr + 1, Nt+1);
 Au = new_matrix(Nr + 1, Nt+1);
 Ar = new_matrix(Nr + 1, Nt+1);
 Ab = new_matrix(Nr + 1, Nt+1);
 /* 確保できたか確認 */
 if ((b == NULL) || (U == NULL) || (UU == NULL)
     || (Bd == NULL) || (Bl == NULL) || (Bu == NULL)
     || (Ad == NULL) || (Al == NULL) || (Au == NULL)
     || (Ar == NULL) || (Ab == NULL)) {
   fprintf(stderr, "memory allocation error\n");
   exit(1);
```

```
  }

  /* r 方向の係数行列 B を作る */
  for (i = 1; i < Nr; i++) {
    Bd[i] = 1.0 + 2 * lambda_r;
    Bl[i] = - lambda_r * (1.0 - 1.0 / (2 * i));
    Bu[i] = - lambda_r * (1.0 + 1.0 / (2 * i));
  }
  /* 係数行列 B を LU 分解する */
  trilu(Nr - 1, Bl + 1, Bd + 1, Bu + 1);

  /* θ 方向の係数行列 Ai を作り、LU 分解する */
  for (i = 1; i < Nr; i++) {
    ri = i * hr;
    rr = ri * ri;
    coef_a = 1 + 2 * lambda_t / rr;
    coef_b = - lambda_t / rr;
    for (j = 0; j < Nt; j++) {
      Al[i][j] = coef_b; Ad[i][j] = coef_a; Au[i][j] = coef_b;
      Ar[i][j] = 0;
      Ab[i][j] = 0;
    }
    Ar[i][0] = coef_b;
    Ab[i][0] = coef_b;
    ptrilu0(Nt, Al[i], Ad[i], Au[i], Ab[i], Ar[i]);
  }

  /* 初期値 */
  for (i = 0; i <= Nr; i++)
    for (j = 0; j <= Nt; j++)
      U[i][j] = u0(i * hr, j * ht);

  /* GNUPLOT を起動 */
#ifdef USEGNUPLOT
  open_gnuplot();
#endif

  /* 時間サイクルをまわす */
  nMax = rint(Tmax / tau);
  for (n = 0; n < nMax; n++) {

    /* 第 n 段から第 n+1/2 段 */
    /* 原点での値を計算（陽解法）*/
    s = 0;
    for (j = 0; j < Nt; j++)
      s += U[1][j];
    UU[0][0] = 4 * lambda_r * (s / Nt - U[0][0]) + U[0][0];
    for (j = 1; j <= Nt; j++)
      UU[0][j] = UU[0][0];
```

```
/* r 方向を陰的に解く */
for (j = 0; j < Nt; j++) {
  /*境界の代入 */
  UU[Nr][j] = boundary_data(1.0, j * ht, (n + 0.5) * tau);
  /* 差分方程式の右辺既知項を計算（係数を事前に計算しておくと良い？）*/
  for (i = 1; i < Nr; i++) {
    ri = i * hr;
    rr = ri * ri;
    coef_a = 1.0 - 2 * lambda_t / rr;
    coef_b = lambda_t / rr;
    if (j != 0)
      b[i] = coef_a * U[i][j] + coef_b * (U[i][j - 1] + U[i][j + 1]);
    else
      b[i] = coef_a * U[i][j] + coef_b * (U[i][Nt - 1] + U[i][j + 1]);
  }
  /* 右辺に移項する処理 */
  b[1] += lambda_r * (1 - 1.0 / 2) * UU[0][j];
  b[Nr - 1] += lambda_r * (1 + 1.0 / (2 * (Nr - 1))) * UU[Nr][j];
  /* r 方向方程式を解く */
  trisol(Nr - 1, Bl + 1, Bd + 1, Bu + 1, b + 1);
  /* UU に計算結果を残す */
  for (i = 1; i < Nr; i++)
    UU[i][j] = b[i];
}
/* j=N θにも値を入れておく（必要ある？）*/
for (i = 0; i <= Nr; i++)
  UU[i][Nt] = UU[i][0];

/* 第 n+1/2 段から第 n+1 段 */
/* θ方向を陰的に解く */
for (i = 1; i < Nr; i++) {
  /* 差分方程式の右辺既知項を計算（係数を事前に計算しておくと良い？）*/
  ri = i * hr;
  rr = hr / (2 * ri);
  coef_a = 1 - 2 * lambda_r;
  coef_b = lambda_r * (1 + rr);
  coef_c = lambda_r * (1 - rr);
  for (j = 0; j < Nt; j++)
    b[j] = coef_a * UU[i][j] + coef_b * UU[i + 1][j] + coef_c * UU[i - 1][j];

  /*θ方向方程式を解く*/
  ptrisol0(Nt, Al[i], Ad[i], Au[i], Ab[i], Ar[i], b);
  /* U に計算結果を残す */
  for (j = 0; j < Nt; j++)
    U[i][j] = b[j];
  /* j=N θにも値を入れておく（必要ある？）*/
  U[i][Nt] = U[i][0];
}
/* 原点での値を計算（陽解法）*/
```

```
    s = 0;
    for (j = 0; j < Nt; j++)
      s += UU[1][j];
    U[0][0] = 4 * lambda_r * (s / Nt - UU[0][0]) + UU[0][0];
    for (j = 1; j <= Nt; j++)
      U[0][j] = U[0][0];
    /* 境界データの代入 */
    t = (n + 1) * tau;
    for (j = 0; j <= Nt; j++)
      U[Nr][j] = boundary_data(1.0, j * ht, t);

    if ((n + 1) % skip == 0) {
      /* 計算結果（数値データ）の出力 */
      if (output)
        for (i = 0; i <= Nr; i++) {
          for (j = 0; j <= Nt; j++) {
            printf("%2d ", j);
            printf("%.3f,", U[i][j]);
          }
          printf("\n");
        }
      sprintf(label, "t=%g", t);
#ifdef USEGNUPLOT
      disk(Nr, Nt, U, label);
#endif
      /* 誤差を測る */
      if (zentai) {
        MaxError = 0.0;
        for (i = 0; i <= Nr; i++) {
          ri = i * hr;
          for (j = 0; j <= Nt; j++) {
            double e;
            e = fabs(exactu(ri, j * ht, t) - U[i][j]);
            if (e > MaxError)
              MaxError = e;
          }
        }
        printf("n=%d, norm=%e, t=%f, 誤差=%e\n",
               n + 1, maxnorm(Nr + 1, Nt + 1, U), t, MaxError);
      }
      else {
        double ex = exactu(0.0, 0.0, t);
        MaxError = fabs(U[0][0] - ex);
        printf("n=%d, norm=%g, t=%g, exactu=%g , 誤差=%g\n",
               n, maxnorm(Nr + 1, Nt + 1, U), t, ex, MaxError);
      }
    }
  }
  return 0;
```

```
}

#ifdef BESSEL
/* 同次 Dirichlet 境界条件で簡単な厳密解 (Bessel 関数を利用) */
#define mu01    (2.404825557695773)

/* 初期値 */
double u0(double r, double theta)
{
  return j0(mu01 * r);
}

/* 厳密解 */
double exactu(double r, double theta, double t)
{
  return exp(- mu01 * mu01 * t) * j0(mu01 * r);
}

/* 境界値 */
double boundary_data(double r, double theta, double t)
{
  /* 同次 Dirichlet 境界条件 */
  return 0.0;
}
#else
/* 非同次 Dirichlet 境界条件で簡単な厳密解 … 定数解！ */
double u0(double r, double theta)
{
  return 1.0;
}

double exactu(double r, double theta, double t)
{
  return 1.0;
}

double boundary_data(double r, double theta, double t)
{
  return 1.0;
}
#endif

double maxnorm(int m, int n, matrix u)
{
  int i, j, ii, jj;
  double tmpmax, absu;
  ii = 0;
  jj = 0;
  tmpmax = fabs(u[0][0]);
```

```
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      if ((absu = fabs(u[i][j])) > tmpmax) {
        tmpmax = absu;
        ii = i;
        jj = j;
      }
  printf("(i,j)=(%d,%d)", ii, jj);
  return tmpmax;
}
```

第3章　行列のスペクトル半径

3.1　安定性

ここまで、それぞれの差分法での数値実験によって安定性について調べてきた。今度は、行列のスペクトル半径を求めることで安定性を調べていく。

n を固定した時、$\{U_{ij}^n\}_{1\leq i\leq N_r-1, 0\leq j\leq N_\theta-1}$ を適当に並べたものを $\vec{U}^n$ とおく。

安定であるとは、$\|\vec{U}^n\|$ が有界である。すなわち

$$\exists M \in \mathbf{R} \ \ \forall n \in \mathbf{N},\ \|\vec{U}^n\| \leq M$$

が成り立つことをいう。

安定 $\Longleftrightarrow r(A) \leq 1$ かつ $r(A) = 1$ の場合、$|\lambda| = 1$ となる固有値 λ の $Jordan$ 細胞のサイズは1

である。

ここで

$$r(A) \stackrel{\text{def.}}{=} A \text{のスペクトル半径} = \ \max|\lambda| \ (\lambda : A \text{の固有値})$$

である。

$\vec{U}^n$ から $\vec{U}^{n+1}$ を求める差分方程式を

$$\vec{U}^{n+1} = A\vec{U}^n$$

の形で書くことが出来れば、行列 A の固有値を調べることによって安定性が解析できる。前に紹介した差分法で導いた方程式はそれぞれ上の形で表すことが出来る。

3.2　行列をつくる

ここからはそれぞれの差分方程式を $\vec{U}^{n+1} = A\vec{U}^n$ の形にするために行列を作っていくことにする。

3.2.1 陽解法の場合

陽解法の差分方程式は

$$U_{i,j}^{n+1} = [1-2\lambda_r - \frac{\lambda_\theta}{r_i^2}]U_{i,j}^n + \lambda_r[(1+\frac{h_r}{2r_i})U_{i+1,j}^n + (1-\frac{h_r}{2r_i})U_{i-1,j}^n] + \frac{\lambda_\theta}{r_i^2}(U_{i,j+1}^n + U_{i,j-1}^n)$$

で書けた。まず簡単のために

$$d_i = [1-2\lambda_r - \frac{\lambda_\theta}{r_i^2}],\ f_i = \lambda_r + \frac{\lambda_r h_r}{2r_i},\ \ell_i = \lambda_r - \frac{\lambda_r h_r}{2r_i},\ t_i = \frac{\lambda_\theta}{r_i^2}$$

とおく。

次に準備として、以下の行列を作る。

$$a = \begin{pmatrix} d_1 & f_1 & & \\ \ell_1 & d_2 & \ddots & \\ & \ddots & \ddots & f_{N_r-2} \\ & & \ell_{N_r-1} & d_{N_r-1} \end{pmatrix}, b = \begin{pmatrix} t1 & & \\ & \ddots & \\ & & t_{N_r-1} \end{pmatrix} \in M(N_r-1;\mathbf{R})$$

$$J = \begin{pmatrix} 0 & 1 & & 1 \\ 1 & 0 & \ddots & \\ & \ddots & \ddots & 1 \\ 1 & & 1 & 0 \end{pmatrix},\ I = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \in M(N_\theta;\mathbf{R})$$

そして Kronecker 積を用いて、行列 B を

$$B = J \otimes b + I \otimes a$$

と定める。これを詳しく書くと

$$
B = \left(\begin{array}{cccc|cccc|cccc|cccc}
d_1 & f_1 & & & t_1 & & & & & & & & t_1 & & & \\
\ell_2 & d_2 & \ddots & & & t_2 & & & \cdots & & \cdots & & & t_2 & & \\
 & \ddots & \ddots & f_{n-1} & & & \ddots & & & & & & & & \ddots & \\
 & & \ell_n & d_n & & & & t_n & & & & & & & & t_n \\
\hline
t_1 & & & & d_1 & f_1 & & & t_1 & & & & & & & \\
 & t_2 & & & \ell_2 & d_2 & \ddots & & & t_2 & & & & & & \\
 & & \ddots & & & \ddots & \ddots & f_{n-1} & & & \ddots & & & & & \\
 & & & t_n & & & \ell_n & d_n & & & & t_n & & & & \\
\hline
 & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & \\
 & & \cdots & & & & \cdots & & & & \cdots & & & & \cdots & \\
 & & & & & & & & & & & & & & & \\
\hline
 & & & & & & & & t_1 & & & d_1 & f_1 & & & t_1 \\
 & & & & & & & & & t_2 & & \ell_2 & d_2 & \ddots & & \\
 & & & & & \cdots & & & & & \ddots & & \ddots & \ddots & f_{n-1} & \\
 & & & & & & & & & & t_n & & & \ell_n & d_n & & t_n \\
\hline
t_1 & & & & & & & & & & & t_1 & & & d_1 & f_1 \\
 & t_2 & & & & \cdots & & & & & & & t_2 & & \ell_2 & d_2 & \ddots \\
 & & \ddots & & & & & & & & & & & \ddots & & \ddots & \ddots & f_{n-1} \\
 & & & t_n & & & & & & & & & & & t_n & & \ell_n & d_n
\end{array}\right)
$$

上のような形になる。ただし、$n = N_r - 1$ とおいた。Kronecker 積を用いているので、対角部分は行列 a が N_θ 個並ぶ。つまり行列 a, b が一つの j に対応している。$(j = 0, \cdots, N_\theta - 1)$

よって行列 B のサイズは $(N_r - 1) \times N_\theta$ となる。

そして最終的に求める行列 A は

$$A = \left(\begin{array}{c|c} 1-4\lambda_r & c \\ \hline 0 & \\ \vdots & B \\ 0 & \end{array}\right)$$

このように書ける。A のサイズは、（B のサイズ＋１）になる。
第１行の成分は原点についての差分方程式

$$U_{0,0}^{n+1} = (1-4\lambda_r)U_{0,0}^{n} + \frac{4\lambda_r}{N_\theta}\sum_{j=0}^{N_\theta-1} U_{1,j}^{n} \qquad (j = 0, 1..N_\theta - 1; n = 0, 1, 2\dots)$$

から書くことができる。$(1,1)$ 成分は右辺第 1 項から $1-4\lambda_r$ とする。次に c の部分は

$$c = (\frac{4\lambda_r}{N_\theta}\ 0\dots 0\ \frac{4\lambda_r}{N_\theta}\ 0\dots 0\dots\dots)$$

と書ける。

以上より、差分方程式から行列 A を求めて $\vec{U}^{n+1} = A\vec{U}^{n}$ の形に表せることが出来た。

3.2.2 半陰スキームの場合

差分方程式は

$$\left(1+\frac{2\lambda_\theta}{r_i^2}\right)U_{i,j}^{n+1} - \frac{\lambda_\theta}{r_i^2}\left(U_{i,j+1}^{n+1} + U_{i,j-1}^{n+1}\right)$$
$$= (1-2\lambda_r)U_{i,j}^{n} + \lambda_r\left[\left(1+\frac{h_r}{2r_i}\right)U_{i+1,j}^{n} + \left(1-\frac{h_r}{2r_i}\right)U_{i-1,j}^{n}\right]$$
$$(i = 1, 2, \cdots, N_r - 1; j = 0, 1, \cdots, N_\theta - 1; n = 0, 1, 2, \cdots)$$

であった。

左辺について

$$d_i = 1 + \frac{2\lambda_\theta}{r_i^2}, \quad t_i = -\frac{\lambda_\theta}{r_i^2}$$

とおく。準備として

$$a = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_{N_r-1} \end{pmatrix}, b = \begin{pmatrix} t1 & & \\ & \ddots & \\ & & t_{N_r-1} \end{pmatrix} \in M(N_r - 1; \mathbf{R})$$

$$J = \begin{pmatrix} 0 & 1 & & 1 \\ 1 & 0 & \ddots & \\ & \ddots & \ddots & 1 \\ 1 & & 1 & 0 \end{pmatrix},\ I = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \in M(N_\theta; \mathbf{R})$$

とおき、陽解法の時と同様にして求めた行列を

$$C = \left(\begin{array}{c|c} 1-4\lambda_r & c \\ \hline 0 & \\ \vdots & B \\ 0 & \end{array}\right),\ c = (\frac{4\lambda_r}{N_\theta}\ 0 \ldots 0\ \frac{4\lambda_r}{N_\theta}\ 0 \ldots 0 \ldots\ldots)$$

とする。

右辺について

$$d_i = 1 - 2\lambda_r,\ f_i = \lambda_r + \frac{\lambda_r h_r}{2r_i},\ \ell_i = \lambda_r - \frac{\lambda_r h_r}{2r_i}$$

とする。準備として

$$a = \begin{pmatrix} d_1 & f_1 & & \\ \ell_1 & d_2 & \ddots & \\ & \ddots & \ddots & f_{N_r-2} \\ & & \ell_{N_r-1} & d_{N_r-1} \end{pmatrix} \in M(N_r - 1; \mathbf{R})$$

$$I = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \in M(N_\theta; \mathbf{R})$$

と行列をつくり、Kronecker 積を用いて

$$B = I \otimes a$$

とすれば、後は同様にして行列

$$D = \left(\begin{array}{c|c} 1-4\lambda_r & c \\ \hline 0 & \\ \vdots & B \\ 0 & \end{array}\right),\ c = (\frac{4\lambda_r}{N_\theta}\ 0 \ldots 0\ \frac{4\lambda_r}{N_\theta}\ 0 \ldots 0 \ldots\ldots)$$

が書ける。

以上より

$$C\vec{U}^{n+1} = D\vec{U}^{n}$$

と表すことが出来て、これは結局

$$\vec{U}^{n+1} = C^{-1}D\vec{U}^{n} = A'\vec{U}^{n}$$

と書ける。

3.2.3 ADI 法の場合

第 n 段から第 $n+1/2$ 段

差分方程式は

$$\left(1+\frac{2\lambda'_\theta}{r_i^2}\right)U_{i,j}^{n+\frac{1}{2}} - \frac{\lambda'_\theta}{r_i^2}(U_{i,j+1}^{n+\frac{1}{2}} + U_{i,j-1}^{n+\frac{1}{2}})$$
$$= (1-2\lambda'_r)U_{i,j}^{n} + \lambda'_r\left[\left(1+\frac{h_r}{2r_i}\right)U_{i+1,j}^{n} + \left(1-\frac{h_r}{2r_i}\right)U_{i-1,j}^{n}\right]$$
$$(i=1,2,\cdots,N_r-1; j=0,1,\cdots,N_\theta-1; n=0,1,\cdots)$$

これは半陰スキームと同じ形であるから先程のように行列を作ることが出来る。

第 $n+1/2$ 段から第 $n+1$ 段

差分方程式は

$$(1+2\lambda'_r)U_{i,j}^{n+1} - \lambda'_r\left[\left(1+\frac{h_r}{2r_i}\right)U_{i+1,j}^{n+1} + \left(1-\frac{h_r}{2r_i}\right)U_{i-1,j}^{n+1}\right]$$
$$= \left(1-\frac{2\lambda'_\theta}{r_i^2}\right)U_{i,j}^{n+\frac{1}{2}} + \frac{\lambda'_\theta}{r_i^2}(U_{i,j+1}^{n+\frac{1}{2}} + U_{i,j-1}^{n+\frac{1}{2}})$$

左辺について

$$d_i = 1+2\lambda'_r,\ f_i = -\lambda'_r - \frac{\lambda'_r h_r}{2r_i},\ \ell_i = -\lambda'_r + \frac{\lambda'_r h_r}{2r_i}$$

とおく。準備として

$$a = \begin{pmatrix} d_1 & f_1 & & \\ \ell_1 & d_2 & \ddots & \\ & \ddots & \ddots & f_{N_r-2} \\ & & \ell_{N_r-1} & d_{N_r-1} \end{pmatrix} \in M(N_r-1;\mathbf{R})$$

$$I = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \in M(N_\theta; \mathbf{R})$$

とおき、Kronecker 積を用いて

$$B = I \otimes a$$

とすれば、後は同様にして行列

$$F = \left(\begin{array}{c|c} 1-4\lambda_r & c \\ \hline 0 & \\ \vdots & B \\ 0 & \end{array}\right),\ c = (\frac{4\lambda_r}{N_\theta}\ 0 \dots 0\ \frac{4\lambda_r}{N_\theta}\ 0 \dots 0 \dots\dots)$$

と書ける。

右辺について

$$d_i = 1 - \frac{2\lambda'_\theta}{r_i^2},\ t_i = \frac{\lambda'_\theta}{r_i^2}$$

とおく。次に

$$a = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_{N_r-1} \end{pmatrix}, b = \begin{pmatrix} t1 & & \\ & \ddots & \\ & & t_{N_r-1} \end{pmatrix} \in M(N_r - 1; \mathbf{R})$$

$$J = \begin{pmatrix} 0 & 1 & & 1 \\ 1 & 0 & \ddots & \\ & \ddots & \ddots & 1 \\ 1 & & 1 & 0 \end{pmatrix},\ I = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \in M(N_\theta; \mathbf{R})$$

として、Kronecker 積を用いて

$$B = J \otimes b + I \otimes a$$

とおけば、後は陽解法の時と同様にして次の行列

$$G = \left(\begin{array}{c|c} 1-4\lambda_r & c \\ \hline 0 & \\ \vdots & B \\ 0 & \end{array}\right),\ c = (\frac{4\lambda_r}{N_\theta}\ 0 \dots 0\ \frac{4\lambda_r}{N_\theta}\ 0 \dots 0 \dots\dots)$$

と書ける。
以上から

$$F\vec{U}^{n+1} = G\vec{U}^{n}$$

の形で表されるが、これも結局

$$\vec{U}^{n+1} = F^{-1}G\vec{U}^{n} = A"\vec{U}^{n}$$

と書ける。
これでそれぞれの行列から固有値が求められるようになった。

3.3 実験結果

まずこれまで説明した、行列を作ってスペクトル半径を求めるプログラムを紹介する。以下は、MATLAB で作ったプログラムである。MATLAB 上では

$$N_r, N_\theta\text{を指定} \Rightarrow \text{プログラムの実行} \Rightarrow \tau\text{を入力して実験}$$

といった流れである。

陽解法の場合

```
hr=1/Nr;ht = 1/Nt;
tau = input(' tau= ');
lambda_r=tau/hr^2;lambda_t=tau/ht^2;
r=hr:hr:(Nr-1)/Nr;
s=1/Nr:1/Nr:(Nr-2)/Nr;
ri = lambda_r + (lambda_r*hr)./2*s;
u=2/Nr:1/Nr:(Nr-1)/Nr;
li = lambda_r -(lambda_r*hr)./2*u;
di=1-2*lambda_r-(2*lambda_t./(r.*r));
ti = lambda_t./(r.*r);
a = diag(di)+diag(ri,1) + diag(li,-1);
b = diag(ti);
J =diag(ones((Nt-1),1),1) + diag(ones((Nt-1),1),-1)
            +diag(1,Nt-1)+diag(1,-(Nt-1));
I = eye(Nt);
B = kron(J,b) + kron(I,a);
A = zeros((Nr-1)*Nt+1); A(2:(Nr-1)*Nt+1,2:(Nr-1)*Nt+1) = B;
c = zeros(1,(Nr-1)*Nt+1);
c(1,1) = 1 - 4 * lambda_r;
for i = 0:Nt-1;
      c(1,(Nr-1)*i+2) = 4 * lambda_r/Nt;
```

```
end;
A(1,1:(Nr-1)*Nt+1) = c;
opts.disp = 0;
abs(eigs(A,1,'lm',opts))
```

半陰スキームの場合

```
hr = 1/Nr; ht = 1/Nt;
tau = input('tau= ');
lambda_r = tau/hr^2; lambda_t = tau/ht^2;
I = eye(Nt);
J = diag(ones(Nt-1,1),1)+diag(ones(Nt-1,1),-1)+diag(1,Nt-1)+diag(1,-Nt+1);
r = 1/Nr:1/Nr:(Nr-1)/Nr;
di = 1 + 2*lambda_t./(r.*r);
ti = -lambda_t./(r.*r);
a = diag(di); b = diag(ti);
B = kron(I,a) + kron(J,b);
A = zeros((Nr-1)*Nt+1); A(2:(Nr-1)*Nt+1,2:(Nr-1)*Nt+1) = B;
c = zeros(1,(Nr-1)*Nt+1); c(1,1) = 1-4*lambda_r;
for i = 0:Nt-1;
    c(1,(Nr-1)*i+2) = 4*lambda_r/Nt;
end;
A(1,1:(Nr-1)*Nt+1) = c;

fi = 1-2*lambda_r;
s = 1/Nr:1/Nr:(Nr-2)/Nr;
ri = lambda_r + (lambda_r*hr)./2*s;
u = 2/Nr:1/Nr:(Nr-1)/Nr;
li = lambda_r - (lambda_r*hr)./2*u;
d = fi * eye(Nr-1) + diag(ri,1) + diag(li,-1);
C = kron(I,d);
D = zeros((Nr-1)*Nt+1); D(2:(Nr-1)*Nt+1,2:(Nr-1)*Nt+1) = C;
g = zeros(1,(Nr-1)*Nt+1); g(1,1) = 1-4*lambda_r;
for j = 0:Nt-1;
    g(1,(Nr-1)*j+2) = 4*lambda_r/Nt;
end;
D(1,1:(Nr-1)*Nt+1) = g;
F = inv(A)*D;
opts.disp = 0;
abs(eigs(F,1,'lm',opts))
```

ADI 法の場合 (r 方向と θ 方向を掛け合わせたもの)

```
hr = 1/Nr; ht = 1/Nt;
```

```
tau = input('tau= ');
lambda_r = tau/hr^2; lambda_t = tau/ht^2;
lambda1_r=lambda_r/2; lambda1_t = lambda_t/2;
I = eye(Nt);
J = diag(ones(Nt-1,1),1)+diag(ones(Nt-1,1),-1)+diag(1,Nt-1)+diag(1,-Nt+1);
r = 1/Nr:1/Nr:(Nr-1)/Nr;
di = 1 + 2*lambda1_t./(r.*r);
ti = -lambda1_t./(r.*r);
a = diag(di); b = diag(ti);
B = kron(I,a) + kron(J,b);
A = zeros((Nr-1)*Nt+1); A(2:(Nr-1)*Nt+1,2:(Nr-1)*Nt+1) = B;
c = zeros(1,(Nr-1)*Nt+1); c(1,1) = 1-4*lambda1_r;
for i = 0:Nt-1;
    c(1,(Nr-1)*i+2) = 4*lambda1_r/Nt;
end;
A(1,1:(Nr-1)*Nt+1) = c;

fi = 1-2*lambda1_r;
s = 1/Nr:1/Nr:(Nr-2)/Nr;
ri = lambda1_r + (lambda1_r*hr)./2*s;
u = 2/Nr:1/Nr:(Nr-1)/Nr;
li = lambda1_r - (lambda1_r*hr)./2*u;
d = fi * eye(Nr-1) + diag(ri,1) + diag(li,-1);
C = kron(I,d);
D = zeros((Nr-1)*Nt+1); D(2:(Nr-1)*Nt+1,2:(Nr-1)*Nt+1) = C;
g = zeros(1,(Nr-1)*Nt+1); g(1,1) = 1-4*lambda1_r;
for j = 0:Nt-1;
    g(1,(Nr-1)*j+2) = 4*lambda1_r/Nt;
end;
D(1,1:(Nr-1)*Nt+1) = g;
F = inv(A)*D;

di = 1 + 2*lambda1_r;
s = 1/Nr:1/Nr:(Nr-2)/Nr;
ri = -lambda1_r - lambda1_r * hr./2*s;
u = 2/Nr:1/Nr:(Nr-1)/Nr;
li = -lambda1_r + lambda1_r * hr./2*u;
a = di*eye(Nr-1) + diag(ri,1) + diag(li,-1);
G = kron(I,a);
H = zeros((Nr-1)*Nt+1); H(2:(Nr-1)*Nt+1,2:(Nr-1)*Nt+1) = G;
b = zeros(1,(Nr-1)*Nt+1);
b(1,1) = 1 - 4 * lambda1_r;
for i = 0:Nt-1;
    b(1,(Nr-1)*i+2) = 4*lambda1_r/Nt;
end;
H(1,1:(Nr-1)*Nt+1) = b;

r = 1/Nr:1/Nr:(Nr-1)/Nr;
fi = 1 - 2*lambda1_t./(r.*r);
```

```
ti = lambda1_t./(r.*r);
c = diag(fi);
d = diag(ti);
K = kron(I,c) + kron(J,d);
L = zeros((Nr-1)*Nt+1); L(2:(Nr-1)*Nt+1,2:(Nr-1)*Nt+1) = K;
g = zeros(1,(Nr-1)*Nt+1);
g(1,1) = 1 - 4 * lambda1_r;
for j = 0:Nt-1;
    g(1,(Nr-1)*j+2) = 4*lambda1_r/Nt;
end;
L(1,1:(Nr-1)*Nt+1) = g;

M = inv(H)*L;
N = F * M;
opts.disp = 0;
abs(eigs(N,1,'lm',opts))

eigs(N,2,'lm',opts)
```

これらのプログラムを使って、安定性を調べるためにτを小さくしていき、$r(A)\leq 1$となる時のτの値を求めた。その結果を以下に記す。

$N_r = 10,\ N_\theta = 20$	
陽解法	$\tau \leq$ 1.2484e-05
半陰	$\tau \leq 0.00513$
ADI (1)θ 方向	$\tau \leq 0.01026$
ADI (2)r 方向	$\tau \leq$ 2.5031e-05
ADI (1)*(2)	$\forall\tau$

$N_r = 20,\ N_\theta = 40$	
陽解法	$\tau \leq$ 7.8102e-07
半陰	$\tau \leq 0.001258$
ADI (1)θ 方向	$\tau \leq 0.002516$
ADI (2)r 方向	$\tau \leq$ 1.563e-06
ADI (1)*(2)	$\forall\tau$

この結果を見ると、陽解法の場合は、数値実験の時よりもτの値を小さくとらなければならなかった。半陰スキーム、ADI 法の場合は前に予想した安定条件と非常に近い値になった。また ADI(2) から、r 方向にのみ陰的に扱う方法は陽解法並みに条件が厳しいことが分かる。

さて、ADI 法でr 方向とθ 方向を掛け合わせた行列のスペクトル半径を調べたところ、任意のτ に対して 1 以下という結果になった。しかし、前の数値実験では、あるτ の値までは不安定という結果が出ているので、矛盾した結果ということになる。このことについてはまだ分かっていない。どちらかのプログラムが間違っているという可能性もあるが、検討する必要がある。

第4章　円柱領域における差分法

ここからは3次元の熱方程式を扱う。

4.1　ADI法

4.1.1　円柱座標への変換

半径 R、高さ H の円柱 $\Omega := \{(x,y,z); x^2+y^2 < R^2, 0 < z < H\}$ で熱方程式の *Dirichlet* 境界値問題を解く。

円柱座標

$$x = r\cos\theta, \quad y = r\sin\theta, \quad z = z$$

を導入すると、Ω に対応するのは

$$\widetilde{\Omega} := \{(r,\theta,z); 0 \le r < R,\ 0 \le \theta < 2\pi,\ 0 < z < H\}$$

である。よって熱方程式は

$$\begin{aligned}
&\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} + \frac{\partial^2 u}{\partial z^2} \qquad ((r,\theta,z) \in \widetilde{\Omega}, t > 0)\\
&u(r,\theta,z,t) = b(r,\theta,z,t) \qquad ((r,\theta,z) \in \partial\widetilde{\Omega}, t > 0)\\
&u(r,\theta,z,0) = u_0(r,\theta,z) \qquad ((r,\theta,z) \in \overline{\widetilde{\Omega}})
\end{aligned}$$

となる。ここで $\widetilde{\Omega}$ の境界は底面、上面、側面の合併であるから

$$\partial\widetilde{\Omega} = \Gamma_b \cup \Gamma_t \cup \Gamma_s$$

ただし

$$\begin{aligned}
\Gamma_s &:= \{(R,\theta,z);\ \theta \in [0,2\pi),\ z \in (0,H)\}\\
\Gamma_b &:= \{(r,\theta,0);\ r \in [0,R],\ \theta \in [0,2\pi)\}\\
\Gamma_t &:= \{(r,\theta,H);\ r \in [0,R],\ \theta \in [0,2\pi)\}
\end{aligned}$$

4.1.2 格子点

分割数 $N_r, N_\theta, N_z \in \mathbf{N}$ と $\tau > 0$ を選ぶ。

$$h_r := \frac{R}{N_r}, \quad h_\theta := \frac{2\pi}{N_\theta}, \quad h_z := \frac{H}{N_z}$$

また

$$\lambda_r := \frac{\tau/3}{h_r^2}, \quad \lambda_\theta := \frac{\tau/3}{h_\theta^2}, \quad \lambda_z := \frac{\tau/3}{h_z^2}$$

とおく。

$r\theta z$ 空間の格子点の座標 (r_i, θ_j, z_k) は

$$r_i := ih_r \quad (0 \le i \le N_r), \quad \theta_j := jh_\theta \quad (0 \le j \le N_\theta - 1), \quad z_k := kh_z \quad (0 \le k \le N_z)$$

で定める。

時刻については

$$t_\ell := \ell\tau$$

であるが、ADI 法を採用するため、ℓ は非負整数 n だけでなく、$n + 1/3, n + 2/3$ という（分母が 3 の整数である）半端な数も許す。

目標は $u(r_i, \theta_j, z_k, t_\ell)$ の近似値 U_{ijk}^ℓ を求めること：

$$U_{ijk}^\ell \simeq u(r_i, \theta_j, z_k, t_\ell)$$

（後述する）差分方程式について、任意の $k \in \{0, 1, \cdots, N_z\}$ に対して、U_{0jk}^ℓ は j が何であっても $(x, y, z) = (0, 0, z_k)$ という一つの点での値を表しているものであるから、値は同じである：

$$U_{0jk}^\ell = U_{00k}^\ell \quad (1 \le j \le N_\theta - 1)$$

4.1.3 境界条件

まず境界条件 $u(r, \theta, z) = b(r, \theta, z)$ $(r = 1$ または $z = 0$ または $z = 1)$ より

$$\begin{aligned}
U_{N_r jk}^{n+1/3} &= b(R, \theta_j, z_k) \qquad (0 \le j \le N_\theta - 1, 1 \le k \le N_z - 1) \\
U_{ij0}^{n+1/3} &= b(r_i, \theta_j, 0) \qquad (0 \le i \le N_r, 0 \le j \le N_\theta - 1) \\
U_{ijN_z}^{n+1/3} &= b(r_i, \theta_j, H) \qquad (0 \le i \le N_r, 0 \le j \le N_\theta - 1)
\end{aligned}$$

4.1.4 円柱の中心軸（$r=0$）におけるラプラシアンの近似

ここで、中心軸について考える。

$$\triangle u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

これより

$$\begin{aligned}\triangle u(0,0,z_k,t_\ell) &\fallingdotseq \frac{U^\ell_{10k} - 2U^\ell_{00k} + U^\ell_{-1,0k}}{h_r^2} + \frac{U^\ell_{01k} - 2U^\ell_{00k} + U^\ell_{0,-1,k}}{h_r^2} + \frac{U^\ell_{00,k+1} - 2U^\ell_{00k} + U^\ell_{00,k-1}}{h_z^2} \\ &= \frac{4}{h_r^2}\left(\frac{U^\ell_{10k} + U^\ell_{-1,0k} + U^\ell_{01k} + U^\ell_{0,-1,k}}{4} - U^\ell_{00k}\right) + \frac{U^\ell_{00,k+1} - 2U^\ell_{00k} + U^\ell_{00,k-1}}{h_z^2}\end{aligned}$$

のような差分近似が考えられる。
ラプラシアンの回転対称性を考慮すれば

$$\triangle u(0,0,z_k,t_\ell) \fallingdotseq \frac{4}{h_r^2}\left(\frac{1}{N_\theta}\sum_{j=0}^{N_\theta-1} U^\ell_{0jk} - U^\ell_{00k}\right) + \frac{U^\ell_{0,0,k+1} - 2U^\ell_{00k} + U^\ell_{00,k-1}}{h_z^2}$$

また

$$\triangle u(0,0,z_k,t_\ell) \fallingdotseq \frac{U^{\ell+1/3}_{00k} - U^\ell_{00k}}{\tau/3}$$

よって整理すると

$$\begin{aligned}U^{\ell+1/3}_{00k} &= U^\ell_{00k} + \frac{\tau}{3}\left[\frac{4}{h_r^2}\left(\frac{1}{N_\theta}\sum_{j=0}^{N_\theta-1} U^\ell_{0jk} - U^\ell_{00k}\right) + \frac{U^\ell_{00,k+1} - 2U^\ell_{00k} + U^\ell_{00,k-1}}{h_z^2}\right] \\ &= U^\ell_{00k} + 4\lambda_r\left(\frac{1}{N_\theta}\sum_{j=0}^{N_\theta-1} U^\ell_{0jk} - U^\ell_{00k}\right) + \lambda_z(U^\ell_{00,k+1} - 2U^\ell_{00k} + U^\ell_{00,k-1})\end{aligned}$$

と差分方程式が導かれる。

4.1.5 第 n 段から第 $n+1/3$ 段：r 方向に陰的に進める。

二次元の時のように前進差分近似や中心差分近似を用い、特に r 方向に陰的になるようにすると次の差分方程式を得る。

$$\begin{aligned}\frac{U^{n+1/3}_{ijk} - U^n_{ijk}}{\tau/3} &= \frac{U^{n+1/3}_{i+1,jk} - 2U^{n+1/3}_{ijk} + U^{n+1/3}_{i-1,jk}}{h_r^2} + \frac{1}{r_i}\frac{U^{n+1/3}_{i+1,jk} - U^{n+1/3}_{i-1,jk}}{2h_r} \\ &\quad + \frac{1}{r_i^2}\frac{U^n_{i,j+1,k} - 2U^n_{ijk} + U^n_{i,j-1,k}}{h_\theta^2} + \frac{U^n_{ij,k+1} - 2U^n_{ikj} + U^n_{ij,k-1}}{h_z^2} \\ &\quad (1 \le i \le N_r - 1, 0 \le j \le N_\theta - 1, 1 \le k \le N_z - 1)\end{aligned}$$

ただし、$U^n_{i,-1,k}=U^n_{i,N_\theta-1,k}$, $\quad U^n_{i,N_\theta,k}=U^n_{i,0,k}$ と考える。
両辺に $\tau/3$ をかける。その際

$$\frac{\tau}{3}\ \frac{1}{r_i 2h_r}=\frac{\tau/3}{2ih_r^2}=\frac{\lambda_r}{2i}$$

に注意すれば

$$\begin{aligned}U^{n+1/3}_{ijk}-U^n_{ijk} &= \lambda_r(U^{n+1/3}_{i+1,j,k}-2U^{n+1/3}_{ijk}+U^{n+1/3}_{i-1,j,k})+\frac{\lambda_r}{2i}(U^{n+1/3}_{i+1,j,k}-U^{n+1/3}_{i-1,j,k})\\ &\quad+\frac{\lambda_\theta}{r_i^2}(U^n_{i,j+1,k}-2U^n_{ijk}+U^n_{i,j-1,k})+\lambda_z(U^n_{i,j,k+1}-2U^n_{ijk}+U^n_{i,j,k-1})\end{aligned}$$

と書ける。移項して整理すると

$$\begin{aligned}&(1+2\lambda_r)U^{n+1/3}_{ijk}-\lambda_r\left[\left(1-\frac{1}{2i}\right)U^{n+1/3}_{i-1,jk}+\left(1+\frac{1}{2i}\right)U^{n+1/3}_{i+1,jk}\right]\\ &=\left(1-\frac{2\lambda_\theta}{r_i^2}-2\lambda_z\right)U^n_{ijk}+\frac{\lambda_\theta}{r_i^2}(U^n_{i,j+1,k}+U^n_{i,j-1,k})+\lambda_z(U^n_{ij,k+1}+U^n_{ij,k-1})\end{aligned}$$

この連立一次方程式を行列とベクトルで表現する。まず N_r-1 次正方行列 A を

$$A_r:=(1+2\lambda_r)I_{N_r-1}-\lambda_r K_{N_r-1}$$

とおく。ここで

$$I_{N_r-1}:=(N_r-1)\text{ 次単位行列}$$

$$K_{N_r-1}:=\begin{pmatrix}0 & 1+\frac{1}{2\cdot 1} & & & & \\ 1-\frac{1}{2\cdot 2} & 0 & 1+\frac{1}{2\cdot 2} & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1-\frac{1}{2(N_r-2)} & 0 & 1+\frac{1}{2(N_r-2)} & \\ & & & 1-\frac{1}{2(N_r-1)} & 0 & \end{pmatrix}$$

と定め、さらに $0\le j\le N_\theta,\ 1\le k\le N_z-1$ に対して N_r-1 次元ベクトル $\mathbf{U}^{n+1/3}_{jk},\mathbf{b}^n_{jk}$ を

$$\mathbf{U}^{n+1/3}_{jk}:=(U^{n+1/3}_{1jk},U^{n+1/3}_{2jk},\cdots,U^{n+1/3}_{N_r-1,jk})^T,$$

$$\mathbf{b}^n_{jk}:=\begin{pmatrix}\beta_1U^n_{1jk}+\frac{\lambda_\theta}{r_1^2}(U^n_{1,j+1,k}+U^n_{1,j-1,k})+\lambda_z(U^n_{1,j,k+1}+U^n_{1,j,k-1})\\ \vdots\\ \beta_iU^n_{ijk}+\frac{\lambda_\theta}{r_i^2}(U^n_{i,j+1,k}+U^n_{i,j-1,k})+\lambda_z(U^n_{i,j,k+1}+U^n_{i,j,k-1})\\ \vdots\\ \beta_mU^n_{mjk}+\frac{\lambda_\theta}{r_m^2}(U^n_{m,j+1,k}+U^n_{m,j-1,k})+\lambda_z(U^n_{m,j,k+1}+U^n_{m,j,k-1})\end{pmatrix}+\lambda_r\begin{pmatrix}(1-\frac{1}{2})U^{n+1/3}_{0jk}\\ 0\\ \vdots\\ 0\\ (1+\frac{1}{2m})U^{n+1/3}_{N_r,jk}\end{pmatrix}$$

$$m := N_r - 1, \quad \beta_i := 1 - \frac{2\lambda_\theta}{r_i^2} - 2\lambda_z (1 \le i \le N_r - 1)$$

で定めると、

$$A_r \mathbf{U}_{jk}^{n+1/3} = \mathbf{b}_{jk}^n$$

と表せる。尚、$\mathbf{b}_{jk}^n$ の計算で、第 $n+1/3$ 段での値 $U_{0jk}^{n+1/3}, U_{N_r,jk}^{n+1/3}$ が必要になるが、それぞれ

$$U_{0jk}^{n+1/3} = U_{00k}^{n+1/3} = U_{00k}^n + 4\lambda_r \left[\frac{1}{N_\theta} \sum_{j=0}^{N_\theta - 1} U_{0jk}^n - U_{00k}^n \right] + \lambda_z (U_{00,k+1}^n - 2U_{00k}^n + U_{00,k-1}^n),$$

$$U_{N_r,jk}^{n+1/3} = b(R, \theta_j, z_k, t_{n+1/3})$$

で計算すればよい。

4.1.6 第 $n+1/3$ 段から第 $n+2/3$ 段：θ 方向に陰的に進める

やはり空間微分に関しては中心差分近似を用い、特に θ 方向に陰的になるようにすると次の差分方程式を得る。

$$\begin{aligned} \frac{U_{ijk}^{n+2/3} - U_{ijk}^{n+1/3}}{\tau/3} &= \frac{U_{i+1,jk}^{n+1/3} - 2U_{ijk}^{n+1/3} + U_{i-1,jk}^{n+1/3}}{h_r^2} + \frac{1}{r_i} \frac{U_{i+1,jk}^{n+1/3} - U_{i-1,jk}^{n+1/3}}{2h_r} \\ &\quad + \frac{1}{r_i^2} \frac{U_{i,j+1,k}^{n+2/3} - 2U_{ijk}^{n+2/3} + U_{i,j-1,k}^{n+2/3}}{h_\theta^2} + \frac{U_{ij,k+1}^{n+1/3} - 2U_{ijk}^{n+1/3} + U_{ij,k-1}^{n+1/3}}{h_z^2} \\ &\quad (1 \le i \le N_r - 1, 0 \le j \le N_\theta - 1, 1 \le k \le N_z - 1) \end{aligned}$$

前回と同様に分母を払って

$$\begin{aligned} U_{ijk}^{n+2/3} - U_{ijk}^{n+1/3} &= \lambda_r (U_{i+1,jk}^{n+1/3} - 2U_{ijk}^{n+1/3} + U_{i-1,jk}^{n+1/3}) + \frac{\lambda_r}{2i} (U_{i+1,jk}^{n+1/3} - U_{i-1,jk}^{n+1/3}) \\ &\quad + \frac{\lambda_\theta}{r_i^2} (U_{i,j+1,k}^{n+2/3} - 2U_{ijk}^{n+2/3} + U_{i,j-1,k}^{n+2/3}) + \lambda_z (U_{ij,k+1}^{n+1/3} - 2U_{ijk}^{n+1/3} + U_{ij,k-1}^{n+1/3}) \end{aligned}$$

移項して整理すると

$$\begin{aligned} &\left(1 + \frac{2\lambda_\theta}{r_i^2}\right) U_{ijk}^{n+2/3} - \frac{\lambda_\theta}{r_i^2} (U_{i,j+1,k}^{n+2/3} + U_{i,j-1,k}^{n+2/3}) \\ = \; &(1 - 2\lambda_r - 2\lambda_z) U_{ijk}^{n+1/3} + \lambda_r \left[\left(1 + \frac{1}{2i}\right) U_{i+1,jk}^{n+1/3} + \left(1 - \frac{1}{2i}\right) U_{i-1,jk}^{n+1/3} \right] \\ &+ \lambda_z (U_{ij,k+1}^n + U_{ij,k-1}^n) \end{aligned}$$

各 $i \in \{1,2,\cdots,N_r-1\}, k \in \{1,2,\cdots,N_z-1\}$ に対して、まず N_θ 次正方行列 A_i を

$$A_i := \left(1+\frac{2\lambda_\theta}{r_i^2}\right) I_{N_\theta} - \frac{\lambda_\theta}{r_i^2} L_{N_\theta}$$

とする。ここで

$$I_{N_\theta} := N_\theta\text{次単位行列}, \qquad L_{N_\theta} := \begin{pmatrix} 0 & 1 & & & 1 \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ 1 & & & 1 & 0 \end{pmatrix}$$

と定め、N_θ 次元ベクトル $\mathbf{U}_{ik}^{n+2/3}, \mathbf{b}_{ik}^{n+2/3}$ を

$$\mathbf{U}_{ik}^{n+2/3} := \left(U_{i0k}^{n+2/3}, U_{i1k}^{n+2/3}, \cdots, U_{i,N_\theta-1,k}^{n+2/3}\right)^T,$$

$$\mathbf{b}_{ik}^{n+2/3} := \begin{pmatrix} (1-2\lambda_r-2\lambda_z)U_{i0k}^{n+1/3} + \lambda_r\left[\left(1+\frac{1}{2i}\right)U_{i+1,0k}^{n+1/3} + \left(1-\frac{1}{2i}\right)U_{i-1,0k}^{n+1/3}\right] + \lambda_z(U_{i0,k+1}^{n} + U_{i0,k-1}^{n}) \\ \vdots \\ (1-2\lambda_r-2\lambda_z)U_{ijk}^{n+1/3} + \lambda_r\left[\left(1+\frac{1}{2i}\right)U_{i+1,jk}^{n+1/3} + \left(1-\frac{1}{2i}\right)U_{i-1,jk}^{n+1/3}\right] + \lambda_z(U_{i,j,k+1}^{n} + U_{i,j,k-1}^{n}) \\ \vdots \\ (1-2\lambda_r-2\lambda_z)U_{imk}^{n+1/3} + \lambda_r\left[\left(1+\frac{1}{2i}\right)U_{i+1,m,k}^{n+1/3} + \left(1-\frac{1}{2i}\right)U_{i-1,m,k}^{n+1/3}\right] + \lambda_z(U_{i,m,k+1}^{n} + U_{i,m,k-1}^{n}) \end{pmatrix}$$

$$m := N_\theta - 1$$

で定める。このとき

$$A_i \mathbf{U}_{ik}^{n+2/3} = \mathbf{b}_{ik}^{n+1/3}$$

と表せる。

4.1.7　第 $n+2/3$ 段から第 $n+1$ 段：z 方向に陰的に進める

やはり空間微分に関しては中心差分近似を用い、特に z 方向に陰的になるようにすると次の差分方程式を得る。

$$\begin{aligned}\frac{U_{ijk}^{n+1}-U_{ijk}^{n+2/3}}{\tau/3} &= \frac{U_{i+1,jk}^{n+2/3}-2U_{ijk}^{n+2/3}+U_{i-1,jk}^{n+2/3}}{h_r^2}+\frac{1}{r_i}\frac{U_{i+1,jk}^{n+2/3}-U_{i-1,jk}^{n+2/3}}{2h_r}\\ &\quad \frac{1}{r_i^2}\frac{U_{i,j+1,k}^{n+2/3}-2U_{ijk}^{n+2/3}+U_{i,j-1,k}^{n+2/3}}{h_\theta^2}+\frac{U_{ij,k+1}^{n+1}-2U_{ijk}^{n+1}+U_{ij,k-1}^{n+1}}{h_z^2}\\ &\quad (1\le i\le N_r-1, 0\le j\le N_\theta-1, 1\le k\le N_z-1)\end{aligned}$$

分母を払って

$$\begin{aligned}U_{ijk}^{n+1}-U_{ijk}^{n+2/3} &= \lambda_r(U_{i+1,jk}^{n+2/3}-2U_{ijk}^{n+2/3}+U_{i-1,jk}^{n+2/3})+\frac{\lambda_r}{2i}(U_{i+1,jk}^{n+2/3}-U_{i-1,jk}^{n+2/3})\\ &\quad +\frac{\lambda_\theta}{r_i^2}(U_{i,j+1,k}^{n+2/3}-2U_{ijk}^{n+2/3}+U_{i,j-1,k}^{n+2/3})+\lambda_z(U_{ij,k+1}^{n+1}-2U_{ijk}^{n+1}+U_{ij,k-1}^{n+1})\end{aligned}$$

移項して整理すると

$$\begin{aligned}&(1+2\lambda_z)U_{ijk}^{n+1}-\lambda_z(U_{ij,k+1}^{n+1}+U_{ij,k-1}^{n+1})\\ =\ &\left(1-2\lambda_r-\frac{2\lambda_\theta}{r_i^2}\right)U_{ijk}^{n+2/3}+\lambda_r\left[\left(1+\frac{1}{2i}\right)U_{i+1,jk}^{n+2/3}+\left(1-\frac{1}{2i}\right)U_{i-1,jk}^{n+2/3}\right]\\ &+\frac{\lambda_\theta}{r_i^2}(U_{i,j+1,k}^{n+2/3}+U_{i,j-1,k}^{n+2/3})\end{aligned}$$

と書ける。

各 $i\in\{1,2,\cdots,N_r-1\}, j\in\{0,1,\cdots,N_\theta-1\}$ に対して、まず N_z-1 次正方行列 A_z を

$$A_z := (1+2\lambda_z)I_{N_z-1}-\lambda_z J_{N_z-1}$$

とおく。ここで

$$I_{N_z-1} := N_z-1\text{ 次の単位行列},\qquad J_{N_z-1} := \begin{pmatrix}0&1&&&\\1&0&1&&\\&\ddots&\ddots&\ddots&\\&&1&0&1\\&&&1&0\end{pmatrix}$$

と定める。

$N_z - 1$ 次元ベクトル $\mathbf{U}_{ij}^{n+1}, \mathbf{b}_{ij}^{n+2/3}$ を

$$\mathbf{U}_{ij}^{n+1} := (U_{ij1}^{n+1}, U_{ij2}^{n+1}, \cdots, U_{ij,N_z-1}^{n+1})^T,$$

$$\mathbf{b}_{ij}^{n+2/3} := \begin{pmatrix} \gamma_i U_{ij1}^{n+2/3} + \lambda_r \left[\left(1 + \frac{1}{2i}\right) U_{i+1,j,1}^{n+2/3} + \left(1 - \frac{1}{2i}\right) U_{i-1,j,1}^{n+2/3} \right] + \frac{\lambda_\theta}{r_i^2} (U_{i,j+1,1}^{n+2/3} + U_{i,j-1,1}^{n+2/3}) \\ \vdots \\ \gamma_i U_{ijk}^{n+2/3} + \lambda_r \left[\left(1 + \frac{1}{2i}\right) U_{i+1,j,k}^{n+2/3} + \left(1 - \frac{1}{2i}\right) U_{i-1,j,k}^{n+2/3} \right] + \frac{\lambda_\theta}{r_i^2} (U_{i,j+1,k}^{n+2/3} + U_{i,j-1,k}^{n+2/3}) \\ \vdots \\ \gamma_i U_{ijm}^{n+2/3} + \lambda_r \left[\left(1 + \frac{1}{2i}\right) U_{i+1,j,m}^{n+2/3} + \left(1 - \frac{1}{2i}\right) U_{i-1,j,m}^{n+2/3} \right] + \frac{\lambda_\theta}{r_i^2} (U_{i,j+1,m}^{n+2/3} + U_{i,j-1,m}^{n+2/3}) \end{pmatrix} + \begin{pmatrix} \lambda_z U_{ij0}^{n+1} \\ 0 \\ \vdots \\ 0 \\ \lambda_z U_{ijN_z}^{n+1} \end{pmatrix}$$

$$m := N_z - 1, \qquad \gamma_i := 1 - 2\lambda_r - \frac{2\lambda_\theta}{r_i^2}$$

で定めると、次のように表せる。

$$A_z \mathbf{U}_{ij}^{n+1} = \mathbf{b}_{ij}^{n+2/3}$$

4.2 終わりに

ここまでが私の研究内容であるが、当初の目標であった円柱領域の差分法のプログラムを作り、数値実験をするというところまで辿り着く事が出来なかったのは残念だった。MATLAB というソフトでの固有値を求める実験に時間がかかってしまい、さらにその結果にも謎が残ってしまったが、それでも興味深い結果が得られたと思う。今回達成できなかったことを後輩に引き継いで取り組んで欲しい。

関連図書

[1] スタンリーファーロウ著, 伊理正夫・伊理由実 訳『偏微分方程式』(朝倉書店)

[2] 藤田 宏『応用数学』(放送大学教育振興会)

[3] 犬井 鉄朗『偏微分方程式とその応用』(コロナ社)

[4] 伊理 正夫『一般線形代数』(岩波書店)

[5] 川久保 勝夫『線形代数学』(日本評論社)

[6] 遠藤 洋一, 高木 章裕, 内藤 達也『円盤領域における熱方程式の研究』(1998 年度卒業研究レポート)