円盤領域における
熱方程式に対する差分法

明治大学 理工学部 数学科
池谷 隆博

2007 年 2 月 26 日

目 次

序

0.1 はじめに

今回、私が研究したテーマは「2 次元円盤領域における熱方程式」である。極座標を導入することにより、$[0,1] \times [0,2\pi]$ という長方形領域での問題になり差分法で計算しやすくなる。

２次元円盤領域において、陽解法でのプログラム (遠藤・高木・内藤 [6]、岡田 [7]) はできているので、その続きとして陰解法で解くことが目標であり、一般化して θ 法で解けるプログラムを作成した。差分解の安定性については、2 次元円盤領域における熱方程式の陽解法の安定性の考察を参考にし、自分なりに予想をたて数値実験を行い検証した。

また、今まで Laplacian の極座標表示

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

に現れる導関数を中心差分商に置き換える近似を用いてきたが、近似の仕方を若干変え Swartztrauber-Sweet(スワルツトラウバー・スイート) 近似 (これより SS 近似と呼ぶ) を用いて、2 次元円盤領域における熱方程式を解くプログラムを作った。そして、その両方の近似の方法を数値実験をして比較した。

0.2 過去の桂田研の探求について

円盤領域における差分法については、桂田研では以下の研究があった。

(1) 1996 年度卒業生 松本 英久 [5] によって、ADI 法によるプログラムが作られた。

(2) 1998 年度卒業生 遠藤 洋一, 高木 章裕, 内藤 達也 [6] によって、陽解法によるプログラムが作られ、安定性条件について

$$\tau \leq \frac{h_r^2 h_\theta^2}{2(1+h_\theta^2)}$$

が考えられた。この安定性条件は τ が非常に小さくなるため厳しい条件である。

他に、角度方向のみ陰的に扱う「半陰スキーム」のプログラムが作られ安定性条件が

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

と予想され、安定性が向上した。

(3) 2004 年度卒業生 岡田 俊宣 [7] によって、ADI 法の安定条件、差分方程式の行列表示が提出された。

0.3 問題

$\Omega = \{(x,y) \in \boldsymbol{R}^2; x^2+y^2 < 1\}$ とし、$\Gamma = \partial\Omega$ とする。このとき、Ω における熱伝導方程式の初期値境界問題を、

$$u_t(x,y,t) = \triangle u(x,y,t) \qquad ((x,y) \in \Omega, t \in (0,\infty)) \tag{1}$$

$$u(x,y,t) = 0 \qquad ((x,y) \in \Gamma, t \in (0,\infty)) \tag{2}$$

$$u(x,y,0) = u_0(x,y) \qquad ((x,y) \in \Omega, t = 0) \tag{3}$$

と設定する。

まずこの問題の厳密解を求めるが、これは省略する。(遠藤・高木・内藤 [6] を見よ。)

この厳密解の公式は次のように与えられる。

熱方程式、境界条件を満たす $u(x,y,t)$ を

$$x = r\cos\varphi,$$

$$y = r\sin\varphi$$

と極座標変換して考え、その厳密解の公式は、

$$\begin{aligned} u(r,\varphi,t) = \frac{1}{2}\sum_{m=0}^{\infty} A_{m0} e^{-\mu_{m,0}^2 t} J_0(\mu_{m,0} r) \\ + \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} e^{-\mu_{m,n}^2 t} J_n(\mu_{m,n} r)(A_{mn}\cos n\varphi + B_{mn}\sin n\varphi) \end{aligned} \tag{4}$$

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方から m 番目の値、$u(r,\varphi,0) = f(r,\varphi)$ として、

$$A_{mn} = \frac{2}{\pi J_{n+1}(\mu_{m,n} r)^2}\int_0^1 r J_n(\mu_{m,n} r)dr \int_{-\pi}^{\pi} f(r,\varphi)\cos n\varphi d\varphi, \tag{5}$$

$$B_{mn} = \frac{2}{\pi J_{n+1}(\mu_{m,n} r)^2}\int_0^1 r J_n(\mu_{m,n} r)dr \int_{-\pi}^{\pi} f(r,\varphi)\sin n\varphi d\varphi \tag{6}$$

である。

次章から差分法による解法と解法のプログラムについて説明していく。

第1章　円盤領域における差分法

1.1　陰解法に入る前にやること

円盤領域における差分法で、陰解法を考える前に知っておかなければならないものがいくつかある。

陽解法の場合と若干同じこと説明してしまうので、「遠藤, 高木, 内藤 [6]」か「桂田 [4]」を参考にしてもらいたい。

1.1.1　2次元における Laplacian の極座標表示

2次元における Laplacian は、

$$\triangle = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{1.1}$$

である。

$$x = r\cos\varphi,$$

$$y = r\sin\varphi$$

とおいて極座標変換すると、

$$\triangle = \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \varphi^2} \tag{1.2}$$

$$(0 < r \leq 1;\quad 0 \leq \varphi \leq 2\pi)$$

となる。

この Laplacian では、分母に r があるため $r = 0$（原点）のときに問題となる。よって、原点では変換前の Laplacian を採用する。

1.1.2　原点以外の点での Laplacian の差分近似

$N_r, N_\varphi \in \boldsymbol{N}$ に対して、

$$h_r := \frac{1}{N_r},\quad h_\varphi := \frac{2\pi}{N_\varphi},$$

$$r_i := ih_r\ \ (i = 0, 1, \ldots, N_r),\ \varphi_j := jh_\varphi\ \ (j = 0, 1, \ldots, N_\varphi),$$

$\tau > 0$ を固定して、

$$t_n := n\tau\ \ (n = 0, 1, 2, \ldots),$$

$$\lambda_r := \frac{\tau}{h_r^2},\ \lambda_\varphi := \frac{\tau}{h_\varphi^2}$$

と定義する。

$$u_{i,j} := u(r_i, \varphi_j)\ \ (i = 0, 1, \ldots, N_r;\ j = 0, 1, \ldots, N_\varphi)$$

として、

(1.2) の各項をそれぞれ中心差分近似をすると、

$$\triangle u(r_i, \varphi_j) = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_r^2} + \frac{1}{r_i}\frac{u_{i+1,j} - u_{i-1,j}}{2h_r} + \frac{1}{r_i^2}\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_\varphi^2} + O(h_r^2 + h_\varphi^2) \tag{1.3}$$

$$(N_r, N_\varphi \to \infty).$$

1.1.3 原点での Laplacian の差分近似

先ほど述べたように、$r = 0$ では Laplacian の表示式 (1.2) は分母に r があるため使えない。そこで、Laplacian の表示式 (1.1) から考える。

$u'_{i,j} = u(x_i, y_j)$ として、原点の $i = 0, j = 0$ を代入し、

$$\triangle u(0,0) = u_{xx}(0,0) + u_{yy}(0,0).$$

次に中心差分近似をとり、

$$\triangle u(0,0) = \frac{u'_{1,0} - 2u'_{0,0} + u'_{-1,0}}{h_x^2} + \frac{u'_{0,1} - 2u'_{0,0} + u'_{0,-1}}{h_y^2} + O(h_x^2 + h_y^2)$$

となる。

$h_x = h_y = h$ として,

$$\triangle u(0,0) = \frac{4}{h^2}[\frac{1}{4}(u'_{1,0} + u'_{-1,0} + u'_{0,1} + u'_{0,-1}) - u'_{0,0}] + O(h^2).$$

次に $h = h_r$ とし、N_φ が 4 の倍数のとき、それぞれの座標を $u_{i,j} = u(r_i, \varphi_j)$ を用いて対応させると、

$$\triangle u(0,0) = \frac{4}{h^2}[\frac{1}{4}(u_{1,0} + u_{1,N_\varphi/4} + u_{1,N_\varphi/2} + u_{1,3N_\varphi/4}) - u_{0,0}] + O(h_r^2)$$

となる。

この式を元に原点の周りで平均を取ると、

$$\triangle u(0,0) = \frac{4}{h_r^2}(\frac{1}{N_\varphi}\sum_{j=0}^{N_\varphi - 1} u_{1,j} - u_{0,0}) + O(h_r^2) \tag{1.4}$$

となる。

1.2 熱方程式の陽解法による差分方程式

目標は時間 $t = n\tau$ での格子上の点 (r_i, φ_j, t_n) における u の値、$u^n_{i,j} = u(r_i, \varphi_j, t_n)$ の近似値 $U^n_{i,j}$ を求めることである。

問題の熱方程式 (1) は極座標変換により、

$$u_t(r,\varphi,t) = u_{rr}(r,\varphi,t) + \frac{1}{r}u_r(r,\varphi,t) + \frac{1}{r^2}u_{\varphi\varphi}(r,\varphi,t). \tag{1.5}$$

陽解法では、左辺の時間微分は前進差分近似する。右辺の空間微分は (1.3) を採用して近似値 $U^n_{i,j}$ について解くと、原点以外の点では、

$$\frac{U^{n+1}_{i,j} - U^n_{i,j}}{\tau} = \frac{U^n_{i+1,j} - 2U^n_{i,j} + U^n_{i-1,j}}{h_r^2} + \frac{1}{r_i}\frac{U^n_{i+1,j} + U^n_{i-1,j}}{2h_r} + \frac{1}{r_i^2}\frac{U^n_{i,j+1} - 2U^n_{i,j} + U^n_{i,j-1}}{h_\varphi^2} \tag{1.6}$$

を得る。

分母を払って移項し、

$$U^{n+1}_{i,j} = (1 - 2\lambda_r - \frac{2\lambda_\varphi}{r_i^2})U^n_{i,j} + \lambda_r[(1 + \frac{h_r}{2r_i})U^n_{i+1,j} + (1 - \frac{h_r}{2r_i})U^n_{i-1,j}] + \frac{\lambda_\varphi}{r_i^2}(U^n_{i,j+1} + U^n_{i,j-1}) \tag{1.7}$$

$$(i = 1, 2, \ldots, N_r - 1; j = 0, 1, \ldots, N_\varphi - 1; n = 0, 1, \ldots).$$

ただし、$U^n_{i,N_\varphi} = U^n_{i,0}, U^n_{i,-1} = U^n_{i,N_\varphi - 1}$ とする。

原点でも、時間微分に前進差分近似、空間微分では (1.4) を採用して近似値をとると、

$$\frac{U^{n+1}_{0,0} - U^n_{0,0}}{\tau} = \frac{4(\frac{1}{N_\varphi}\sum_{j=0}^{N_\varphi - 1} U^n_{1,j} - U^n_{0,0})}{h_r^2}. \tag{1.8}$$

これを同様に整理して、

$$U^{n+1}_{0,0} = (1 - 4\lambda_r)U^n_{0,0} + \frac{4\lambda_r}{N_\varphi}\sum_{j=0}^{N_\varphi - 1} U^n_{1,j} \tag{1.9}$$

$$(j = 0, 1, \ldots, N_\varphi - 1; n = 0, 1, \ldots).$$

$U^{n+1}_{0,j}$ は j によらず共通の値（r=0 の時原点 1 点に対応するので）とすると,

$$U^n_{0,j} = U^n_{0,0} \qquad (j = 0, 1, \ldots, N_\varphi - 1).$$

これが陽解法による差分方程式である。

1.3 熱方程式の陰解法による差分方程式

1.3.1 原点以外の点での差分方程式

原点以外の点に関して、左辺の時間微分を後退差分近似、右辺の空間微分は (1.3) とした差分方程式を作る。

$$\frac{U^n_{i,j} - U^{n-1}_{i,j}}{\tau} = \frac{U^n_{i+1,j} - 2U^n_{i,j} + U^n_{i-1,j}}{h_r^2} + \frac{1}{r_i}\frac{U^n_{i+1,j} - U^n_{i-1,j}}{2h_r} + \frac{1}{r_i^2}\frac{U^n_{i,j+1} - 2U^n_{i,j} + U^n_{i,j-1}}{h_\varphi^2}. \tag{1.10}$$

この式の番号 n を 1 つずつずらすと、

$$\frac{U_{i,j}^{n+1}-U_{i,j}^{n}}{\tau}=\frac{U_{i+1,j}^{n+1}-2U_{i,j}^{n+1}+U_{i-1,j}^{n+1}}{h_r^2}+\frac{1}{r_i}\frac{U_{i+1,j}^{n+1}+U_{i-1,j}^{n+1}}{2h_r}+\frac{1}{r_i^2}\frac{U_{i,j+1}^{n+1}-2U_{i,j}^{n+1}+U_{i,j-1}^{n+1}}{h_\varphi^2}. \tag{1.11}$$

τ を両辺にかけて、$\lambda_r, \lambda_\varphi$ に置き換え、右肩が $n+1$ となっている項を左辺に、右肩が n となっている項を右辺に移項して整理すると、

$$(1+2\lambda_r+\frac{2\lambda_\varphi}{r_i^2})U_{i,j}^{n+1}-\lambda_r[(1+\frac{h_r}{2r_i})U_{i+1,j}^{n+1}+(1-\frac{h_r}{2r_i})U_{i-1,j}^{n+1}]-\frac{\lambda_\varphi}{r_i^2}(U_{i,j+1}^{n+1}-U_{i,j-1}^{n+1})=U_{i,j}^{n} \tag{1.12}$$

$$(i=1,2,\ldots,N_r-1; j=0,1,\ldots,N_\varphi-1; n=0,1,\ldots).$$

円盤は 2π の周期なので、$\varphi=0$ と $\varphi=2\pi$ での近似値 $U_{i,j}^n$ が同じ位置にあるものとして $U_{i,N_\varphi}^{n+1}=U_{i,0}^{n+1}$ と考える。

同様に $U_{i,-1}^{n+1}=U_{i,N_\varphi-1}^{n+1}$ と考える。

1.3.2 原点での差分方程式

原点に関しては、(1.4) を用いて熱方程式を考える。左辺の時間微分を後退差分近似、右辺の空間微分を中心差分近似すると、

$$\frac{U_{0,0}^{n}-U_{0,0}^{n-1}}{\tau}=\frac{4(\frac{1}{N_\varphi}\sum_{j=0}^{N_\varphi-1}U_{1,j}^{n}-U_{0,0}^{n})}{h_r^2} \tag{1.13}$$

番号 n を 1 つずつずらして、

$$\frac{U_{0,0}^{n+1}-U_{0,0}^{n}}{\tau}=\frac{4(\frac{1}{N_\varphi}\sum_{j=0}^{N_\varphi-1}U_{1,j}^{n+1}-U_{0,0}^{n+1})}{h_r^2} \tag{1.14}$$

分母を払って整理すると、

$$(1+4\lambda_r)U_{0,0}^{n+1}-\frac{4\lambda_r}{N_\varphi}\sum_{j=0}^{N_\varphi-1}U_{1,j}^{n+1}=U_{0,0}^{n} \tag{1.15}$$

$$(j=0,1,\ldots,N_\varphi-1; n=0,1,\ldots).$$

$U_{0,j}^{n+1}$ は j によらず共通の値（原点 1 点に対応するので）でなければならないので、

$$U_{0,j}^{n+1}=U_{0,0}^{n+1} \qquad (j=0,1,\ldots,N_\varphi-1).$$

このように (1.12)、(1.15) の陰解法の差分方程式が得られる。この差分方程式と境界条件 (2), 初期条件 (3) の式を使い、解いていけばよい。

次の節で,θ 法について述べていく。

1.4 θ 法

これまで、陽解法と陰解法それぞれについて差分方程式を考えてきたが、陽解法の差分方程式 (1.6),(1.8) と陰解法の差分方程式 (1.11),(1.14) を $1-\theta:\theta$ $(0\leq\theta\leq 1)$ で重みをつけ重ね合わせた式を考える。

(θ に 0 を代入すれば陽解法、1 を代入すれば陰解法になる。)

1.4.1 原点以外の点での差分方程式

(1.6),(1.14) の式に、$(1-\theta)$ とθ の重みをつけて重ね合わせると、

$$\frac{U_{i,j}^{n+1}-U_{i,j}^{n}}{\tau}=(1-\theta)\left[\frac{U_{i+1,j}^{n}-2U_{i,j}^{n}+U_{i-1,j}^{n}}{h_r^2}+\frac{1}{r_i}\frac{U_{i+1,j}^{n}-U_{i-1,j}^{n}}{2h_r}+\frac{1}{r_i^2}\frac{U_{i,j+1}^{n}-2U_{i,j}^{n}+U_{i,j-1}^{n}}{h_\varphi^2}\right]$$

$$+\theta\left[\frac{U_{i+1,j}^{n+1}-2U_{i,j}^{n+1}+U_{i-1,j}^{n+1}}{h_r^2}+\frac{1}{r_i}\frac{U_{i+1,j}^{n+1}-U_{i-1,j}^{n+1}}{2h_r}+\frac{1}{r_i^2}\frac{U_{i,j+1}^{n+1}-2U_{i,j}^{n+1}+U_{i,j-1}^{n+1}}{h_\varphi^2}\right] \tag{1.16}$$

$$(i=1,2,\ldots,N_r-1;j=0,1,\ldots,N_\varphi-1;n=0,1,\ldots)$$

となる。

τ を両辺にかけて $\lambda_r,\lambda_\varphi$ に置き換え、右肩が $n+1$ となっている項を左辺に、n となっている項を右辺に移項し整理すると、

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

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

$$+\frac{(1-\theta)\lambda_\varphi}{r_i^2}(U_{i,j+1}^{n}-U_{i,j-1}^{n}) \tag{1.17}$$

$$(i=1,2,\ldots,N_r-1;j=0,1,\ldots,N_\varphi-1;n=0,1,\ldots).$$

(陰解法の時と同様、

$$U_{i,N_\varphi}^{n+1}=U_{i,0}^{n+1},U_{i,-1}^{n+1}=U_{i,N_{\varphi-1}}^{n+1},$$

$$U_{i,N_\varphi}^{n}=U_{i,0}^{n},U_{i,-1}^{n}=U_{i,N_{\varphi-1}}^{n}$$

と考える。)

1.4.2 原点での差分方程式

原点の場合も同様に (1.8),(1.14) の式に、$(1-\theta)$ とθ の重みをつけて重ね合わせると、

$$\frac{U_{0,0}^{n+1}-U_{0,0}^{n}}{\tau}=(1-\theta)\left[\frac{4(\frac{1}{N_\varphi}\sum_{j=0}^{N_\varphi-1}U_{1,j}^{n}-U_{0,0}^{n})}{h_r^2}\right]+\theta\left[\frac{4(\frac{1}{N_\varphi}\sum_{j=0}^{N_\varphi-1}U_{1,j}^{n+1}-U_{0,0}^{n+1})}{h_r^2}\right] \tag{1.18}$$

$$(j = 0, 1, \ldots, N_\theta - 1; n = 0, 1, \ldots)$$

となり、τ を両辺にかけて $\lambda_r, \lambda_\varphi$ に置き換え、右肩が $n+1$ となっている項を左辺に、n となっている項を右辺に移項し整理すると、

$$(1 + 4\theta\lambda_r)U_{0,0}^{n+1} - \frac{4\theta\lambda_r}{N_\varphi}\sum_{j=0}^{N_\varphi-1} U_{1,j}^{n+1} = (1 - 4(1-\theta)\lambda_r)U_{0,0}^{n} + \frac{4(1-\theta)\lambda_r}{N_\varphi}\sum_{j=0}^{N_\varphi-1} U_{1,j}^{n} \qquad (1.19)$$

$$(j = 0, 1, \ldots, N_\varphi - 1; n = 0, 1, \ldots).$$

$U_{0,j}^n$ は j によらず共通の値（原点 1 点に対応するので）でなければならないので、

$$U_{0,j}^n = U_{0,0}^n, \quad U_{0,j}^{n+1} = U_{0,0}^{n+1} \qquad (j = 0, 1, \ldots, N_\varphi - 1)$$

と考える。

次の節で、この θ 法による差分方程式を用いて解を求めるプログラムを考える。

1.4.3 i 方向番号付けのプログラム

i 方向番号付けとは、(r_i, φ_j) に $jN_r + i$ という番号をつけたプログラムである。

$\ell = jN_r + i,\ m = N_r + 1$、(1.17),(1.19) の差分方程式の右辺を b_ℓ とおく。原点以外の差分方程式 (1.17) は、

$$(1+2\theta\lambda_r + \frac{2\theta\lambda_\varphi}{r_i^2})U_\ell^{n+1} - \theta\lambda_r[(1+\frac{h_r}{2r_i})U_{\ell+1}^{n+1} + (1-\frac{h_r}{2r_i})U_{\ell-1}^{n+1}] - \frac{\theta\lambda_\varphi}{r_i^2}(U_{\ell+m}^{n+1} - U_{\ell-m}^{n+1}) = b_\ell \quad (1.20)$$

$$(i = 1, 2, \ldots, N_r - 1; j = 0, 1, \ldots, N_\varphi - 1; n = 0, 1, \ldots).$$

原点の差分方程式 (1.19) では $i = 0,\ j = 0$ より、

$$(1 + 4\theta\lambda_r)U_0^{n+1} - \frac{4\theta\lambda_r}{N_\varphi}\sum_{j=0}^{N_\varphi-1} U_{jN_r+1}^{n+1} = b_0 \qquad (1.21)$$

$$(j = 0, 1, \ldots, N_\varphi - 1; n = 0, 1, \ldots).$$

原点のコピーとして、

$$U_{jN_r}^n = U_0^n, \quad U_{jN_r}^{n+1} = U_0^{n+1} \qquad (j = 0, 1, \ldots, N_\varphi - 1)$$

とできる。

```
/*
 * heat2d-i-en.c ---円盤状の 2 次元熱方程式を陰解法で解く
 *     i 方向番号付け のプログラム
 *    コンパイルするには
```

```
 *     gcc -c lu.c
 *     gcc -c call_gnuplot.c
 *
 *     ccmg heat2d-i-en.o lu.c call_gnuplot.o
 */
#include <stdio.h>
#include <math.h>
#include <matrix.h>
#include "lu.h"
#include "call_gnuplot.h"
#define  psi(i,j) ((j) * mm + (i))

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

int main()
{
  double ri ,ri2, phi_j;
  int N_r, N_p, mm, NN, i, j, p, q, n, skip, nMax, L;
  matrix Uk, A;
  double *B, *vector_U, cond;
  int *iwork;
  double h_r, h_p, lambda_r, lambda_p, lambda, tau, t, Tmax, dt, M, ex;
  double theta;
  char label[200];

  /* πの値*/
  pi = 4.0 * atan(1.0);

  /* 区間の分割数 */
   printf("Nr, Ntheta: "); scanf("%d %d", &N_r, &N_p);

  mm = N_r+1;
  NN = (N_r+1)*(N_p);

  /* 空間の刻み幅 */
  h_r = 1.0 / N_r;
```

```
h_p = 2.0 * pi / N_p;


/* 行列、ベクトルを記憶する変数のメモリー割り当て */
  if ((Uk = new_matrix(N_r+1, N_p+1)) == NULL) {
      fprintf(stderr, "数列 U^k を記憶する領域の確保に失敗\n");
      exit(1);
  }
  if ((A = new_matrix(NN, NN)) == NULL) {
      fprintf(stderr, "係数行列 A を記憶する領域の確保に失敗\n");
      exit(1);
  }
  if ((B = (double *) malloc(sizeof(double) * NN)) == NULL) {
      fprintf(stderr, "B を記憶する領域の確保に失敗\n");
      exit(1);
  }
  if ((vector_U = (double *) malloc(sizeof(double) * NN)) == NULL) {
      fprintf(stderr, "vector_U を記憶する領域の確保に失敗\n");
      exit(1);
  }
  if ((iwork = (int *) malloc(sizeof(int) * NN)) == NULL) {
      fprintf(stderr, "iwork を記憶する領域の確保に失敗\n");
      exit(1);
  }

  /* θ法の重みの決定 */
  printf("θ (0≦θ≦1):  "); scanf("%lf", &theta);

  if (theta == 1.0) {
      printf("τ: "); scanf("%lf", &tau);
  } else {
    printf("τ (≦%g         ≡安定性条件): ",
           (h_r * h_r * h_p * h_p) / (2.0 *(1 - theta) * (1 + h_p * h_p)));
     scanf("%lf", &tau);
  }

  /* λr, λθ */
  lambda_r = tau / (h_r * h_r);
  lambda_p = tau / (h_p * h_p);
```

```
    /* 結果を出力する時間間隔の決定 */
    printf("Tmax: "); scanf("%lf", &Tmax);
    printf("Δ t(>=%g): ", tau);  scanf("%lf", &dt);
    if (dt < tau) {
       dt = tau;
    }
    skip = rint(dt / tau);

    /* GNUPLOT の準備 */
    open_gnuplot();

  /* 初期値の設定 */
    for (i = 0; i <= N_r; i++) {
      ri = (i) * h_r;
      for (j = 0; j <= N_p; j++)
        Uk[i][j] = u0(ri, (j) * h_p);
    }
    /* 初期のグラフ */
    disk(N_r, N_p, Uk,"t=0");

    /* 係数行列の値のセット */
    /* 0 クリア */
    for(p = 0; p < NN; p++){
      for(q = 0; q < NN; q++)
        A[p][q] = 0.0;
       A[p][p] = 1.0;
             /* Dirichle 境界条件 原点のコピーのためあらかじめ単位行列を作る */
    }
    /* 原点の係数 */
    A[0][0] = 1.0 + 4.0*theta*lambda_r;

    for(j = 0; j <= N_p-1; j++){
      L =  psi(1,j);
      A[0][L] = - (4.0*theta*lambda_r) / N_p;
      }

/* 原点のコピーの係数 */
    for(j = 1; j <= N_p-1; j++){
      L = psi(0,j);
```

```
    A[L][0] = - 1.0;
  }

/* 内点での係数 */
  for(i = 1; i< N_r; i++){
       ri = (i)*h_r;
       ri2 = ri*ri;
       for(j = 0; j <= N_p-1; j++){
        L = psi(i,j);
        int L1 = L-mm;
        int L2 = L+mm;
        if(j == 0)
          L1 = psi(i,N_p-1);
            /* j=0 の時 j-1 が-1 となるため-1 の代わりに N θ-1 を代入する */
        if(j == N_p-1)
          L2 = psi(i,0);
            /* j=N_p-1 の時 j+1 が N_p となるため N_p の代わりに 0 を代入する */


        A[L][L] = 1.0 + 2.0*theta*lambda_r + (2.0*theta*lambda_p) / ri2;
        A[L][L+1] = - theta* lambda_r * (1.0 + h_r/(2.0*ri));
        A[L][L-1] = - theta* lambda_r * (1.0 - h_r/(2.0*ri));
        A[L][L1] = - theta * lambda_p / ri2;
        A[L][L2] = - theta * lambda_p / ri2;
      }
       }

  /* 係数行列 LU 分解 */
       decomp(NN, A,  &cond, iwork, B);
       if(cond + 1 == cond){
         printf("MATRIX IS SINGULAR TO WORKING PRECISION\n");
         return 0;
       }

  /* 時間に関するループ */
  nMax = rint(Tmax / tau);
  for (n = 1; n <= nMax; n++) {
    /* 連立 1 次方程式の右辺を用意する */
    /* 内部の格子点 */
```

```
      for (i = 1; i < N_r; i++){
      ri = (i)*h_r;
      ri2 = ri*ri;
      for(j = 0; j<= N_p-1; j++){
          L = psi(i,j);
          int jm1 = j-1;
          int jm2 = j+1;
      if(jm1 == -1)
            jm1 = N_p-1;
      if(jm2 == N_p)
            jm2 = 0;
          B[L] =  (1.0 - 2.0*(1.0 - theta)*lambda_r
                - (2.0*(1.0 - theta)*lambda_p)/ri2)*Uk[i][j]
                + (1.0 - theta)*lambda_r*((1.0 + h_r/(2.0*ri))*Uk[i+1][j]
                + (1.0 - h_r/(2.0*ri))*Uk[i-1][j])
                + ((1.0 - theta)*lambda_p / ri2)*(Uk[i][jm1] + Uk[i][jm2]);
      }
      }

      /* 原点 */
      B[0] = (1.0 - 4.0*(1.0 - theta)*lambda_r)*Uk[0][0];
    for(j = 0; j <= N_p-1; j++){
      B[0] += (4.0*(1.0 - theta)*lambda_r / N_p)*Uk[1][j];
    }

    /* 原点のコピー */
    for(j = 1; j <= N_p-1; j++){
      L = psi(0,j);
      B[L] = 0.0;
    }

    /* 境界条件 */
    for(j = 0; j <= N_p-1; j++){
      L = psi(N_r,j);
      B[L] = 0.0;
    }

/* A vector_U = B を解く */
    solve(NN, A, B, iwork);
```

```
 /* 更新 */
   for(i = 0; i<= N_r; i++){
     for(j = 0; j <= N_p-1; j++){
       L = psi(i,j);
       Uk[i][j] = B[L];
     }
     Uk[i][N_p]= B[psi(i,0)];
   }
 t = n * tau;

 /* 誤差を測る */
   M = 0.0;
   for(i = 0; i <= N_r; i++){
     ri = (i)*h_r;
     for(j = 0; j <= N_p; j++){
       double e;
       phi_j = (j)*h_p;
       e = fabs(exactu(ri, phi_j, t) - Uk[i][j]);
       if(e > M)
         M = e;
     }
   }
   printf("n=%d, norm=%g, t=%g, 誤差=%g\n",
          n, maxnorm(N_r, N_p, Uk), t, M);

   /* Δ t の整数倍の時刻ではグラフをを描く */
   if (n % skip == 0){
     sprintf(label, "t=%g", t);
     disk(N_r, N_p, Uk, label);
   }
}
  close_gnuplot();

 return 0;
}
```

```
 /* 厳密解を計算する関数 */

#define mu01    (2.404825557695771998)
#define mu11    (3.831705970207510692)

     /* 初期データ */
double u0(double r, double phi)
  {
    return j0(mu01 * r) + j1(mu11 * r) * (cos(phi) + sin(phi));
  }

    /* 厳密解 */
double exactu(double r, double phi, double t)
          {
            return exp(- mu01* mu01* t)* j0(mu01 * r)
                 + exp(- mu11* mu11* t)* j1(mu11 * r)*(cos(phi) + sin(phi));
          }

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

1.4.4　効率を考えたプログラム (j 方向番号付け)

前小節に書いたプログラムは計算する分には十分であるが、N_r,N_φ を大きくすると計算量が増えて解を計算するのに時間がかかってしまう。そこで、効率良く計算するためにプログラムを改良する必要がある。

計算量を少しでも減らすために、$U_{i,j}$ を i 方向に番号付けをした連立方程式を

$$AU = B,$$

$$A \text{ は係数行列}, U = \begin{pmatrix} U_0^{n+1} \\ U_1^{n+1} \\ \vdots \\ U_N^{n+1} \end{pmatrix} B = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{pmatrix}$$

と表す。

係数行列 A に注目し、$N_r = N_1, N_\varphi = N_2$ $(N_1, N_2 \in \boldsymbol{N})$ として考えていく。

まず、差分方程式（1.17）の係数を、

$$a_i = [1 + 2\theta\lambda_r + \frac{2\theta\lambda_\varphi}{r_i^2}],\quad c_i = -\theta\lambda_r(1 - \frac{h_r}{2r_i}),\quad d_i = -\theta\lambda_r(1 + \frac{h_r}{2r_i}),\quad t_i = -\frac{\theta\lambda_\varphi}{r_i^2}$$

とおく。

次に、$(N_r + 1)$ 次正方行列 P, P', Q, Q', T, T', T'' を

$$P = \begin{pmatrix} 1+4\theta\lambda_r & -\frac{4\theta\lambda_r}{N_\varphi} & & & & \\ c_1 & a_1 & d_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & c_{N_1-1} & a_{N_1-1} & d_{N_1-1} & \\ & & & 0 & 1 & \end{pmatrix},\ P' = \begin{pmatrix} 1 & 0 & & & & \\ c_1 & a_1 & d_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & c_{N_1-1} & a_{N_1-1} & d_{N_1-1} & \\ & & & 0 & 1 & \end{pmatrix},$$

$$Q = \begin{pmatrix} -1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{pmatrix},\ Q' = \left(\begin{array}{c|cccc} 0 & -\frac{4\theta\lambda_r}{N_\varphi} & 0 & \dots & 0 \\ \hline 0 & 0 & & & \\ \vdots & & \ddots & & \\ \vdots & & & \ddots & \\ 0 & & & & 0 \end{array}\right),$$

$$T = \begin{pmatrix} 0 & & & & \\ & t_1 & & & \\ & & \ddots & & \\ & & & t_{n-1} & \\ & & & & 0 \end{pmatrix},\quad T' = Q + T,\quad T'' = Q' + T$$

と定義する。

これらを使い係数行列 A を表すと、

$$A = \begin{pmatrix} P & T'' & Q' & \dots & \dots & Q' & T'' \\ T' & P' & T & & & & \\ Q & T & P' & T & & & \\ \vdots & & \ddots & \ddots & \ddots & & \\ Q & & & & T & P' & T \\ T' & & & & & T'' & P' \end{pmatrix}$$

となる。

帯行列の傾向が見えるが、やはり成分が広く散らばっていて計算量を減らせない。

今度は、$\ell = iN_\varphi + j$(これを j 方向番号付けとする) として係数行列 A を見てみる。

そこで、今度は N_φ 次正方行列 I, J, K, C, D, F, G を

$$I = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix},\ J = \begin{pmatrix} 0 & 1 & & & 1 \\ 1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 1 & & & 1 & 0 \end{pmatrix},$$

$$K_i = a_i I + t_i J,$$

$$C_i = c_i I,$$

$$D_i = d_i I,$$

$$F = \left(\begin{array}{c|ccc} 1+4\theta\lambda_r & 0 & \dots & 0 \\ \hline -1 & & & \\ \vdots & & I & \\ -1 & & & \end{array}\right),\quad G = \left(\begin{array}{ccc} -\frac{4\theta\lambda_r}{N_\varphi} & \dots & -\frac{4\theta\lambda_r}{N_\varphi} \\ \hline & & \\ & \mathbf{0} & \\ & & \end{array}\right)$$

と定義する。

これらを使い係数行列 A を表すと、

$$A = \begin{pmatrix} F & G & & & & \\ C_1 & K_1 & D_1 & & & \\ & C_2 & K_2 & D_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & C_{N_1-1} & K_{N_1-1} & D_{N_1-1} \\ & & & & \mathbf{0} & I \end{pmatrix}$$

となる。

上の係数行列は帯状の行列になり、0 以外の成分が対角線の成分から最も離れた位置にあるのは 0 行目なので、計算する範囲を対角成分から $2N_\varphi - 1$ 離れた成分までにすれば計算量を大幅に少なくできる。

その他に j 方向番号付けの係数行列 A で見てもらうポイントは、一番右下にある行列 I の Dirichlet 境界条件で使われている行と行列 F の 1 行目以下の原点のコピー ($U_{0,j}^{n+1} = U_{0,0}^{n+1}$) に使われている行である。LU 分解をして連立方程式を解くが、ここの行で無駄な計算をしている。

Dirichlet 境界条件から、$U_{N_r,j}$ は既知 ($U_{N_r,j} = 0$) である差分方程式に現れる $U_{N_r,j}$ に代入することによって、上の連立方程式の未知数からはずすことができる。

つまり $i = N_r - 1$ の時左辺を、

$$(1 + 2\theta\lambda_r + \frac{2\theta\lambda_\varphi}{r_i^2})U_{i,j}^{n+1} - \theta\lambda_r(1 - \frac{h_r}{2r_i})U_{i-1,j}^{n+1} - \frac{\theta\lambda_\varphi}{r_i^2}(U_{i,j+1}^{n+1} - U_{i,j-1}^{n+1}) \tag{1.22}$$

のように書き換えればよい。

原点での値 $U_{0,j}^{n+1} (j \geq 1)$ については、$U_{0,0}^{n+1}$ を求めてから代入すればいいのでこれも連立方程式からはずす。また、$U_{0,j}^{n+1}$ が使われている差分方程式では $U_{0,j}^{n+1} = U_{0,0}^{n+1}$ と代入することで解決できる。

プログラム内では $i = 1$ の時左辺を、

$$(1 + 2\theta\lambda_r + \frac{2\theta\lambda_\varphi}{r_i^2})U_{i,j}^{n+1} - \theta\lambda_r[(1 + \frac{h_r}{2r_i})U_{i+1,j}^{n+1} + (1 - \frac{h_r}{2r_i})U_{0,0}^{n+1}]$$

$$-\frac{\theta\lambda_\varphi}{r_i^2}(U_{i,j+1}^{n+1} - U_{i,j-1}^{n+1}) \tag{1.23}$$

と書き換える。

この操作をした係数行列 A を表示すると、

$$A = \left(\begin{array}{c|ccc|cccc} 1 + 4\theta\lambda_r & -\frac{4\theta\lambda_r}{N_\varphi} & \cdots & -\frac{4\theta\lambda_r}{N_\varphi} & 0 & \cdots & \cdots & 0 \\ \hline c_1 & & & & & & & \\ \vdots & & K_1 & & D_1 & & & \\ c_1 & & & & & & & \\ \hline 0 & & C_2 & & K_2 & D_2 & & \\ \vdots & & & & \ddots & \ddots & \ddots & \\ \vdots & & & & & C_{N_1-2} & K_{N_1-2} & D_{N_1-2} \\ 0 & & & & & & C_{N_1-1} & K_{N_1-1} \end{array}\right)$$

となる。

計算する範囲がさらに小さくなり、0 行目を参考にして対角成分から N_φ までの範囲にすればいいことになる。

ここまで連立方程式をはずす操作をしたが、今度はプログラム内においてベクトル U の番号付けがバラバラになってしまう。そのため、新しく番号付けを行う。

まず、原点では番号 0、内点 $(1 \leq i \leq N_r - 1; 0 \leq j \leq N_\varphi - 1)$ は順次 j 方向に番号を付けていくような付け方にすると、原点では常に番号は 0 として内点は $(i-1)N_\varphi + j + 1$ といった順に付けていく。

上の操作をすればいいが、一般化して考えると若干わかりにくいので $N_r = 3, N_\phi = 3$ で見てみよう。(ここでは係数行列の成分をわかりやすくするため τ=1 とした。)

まず、i 方向番号付けでの係数行列は、

$$\left(\begin{array}{cccc|cccc|cccc}
37.0 & -12.0 & 0.0 & 0.0 & 0.0 & -12.0 & 0.0 & 0.0 & 0.0 & -12.0 & 0.0 & 0.0 \\
-4.5 & 23.1 & -13.5 & 0.0 & 0.0 & -2.1 & 0.0 & 0.0 & 0.0 & -2.1 & 0.0 & 0.0 \\
0.0 & -6.8 & 20.0 & -11.3 & 0.0 & 0.0 & -0.5 & 0.0 & 0.0 & 0.0 & -0.5 & 0.0 \\
0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
\hline
-1.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & -2.1 & 0.0 & 0.0 & -4.5 & 23.1 & -13.5 & 0.0 & 0.0 & -2.1 & 0.0 & 0.0 \\
0.0 & 0.0 & -0.5 & 0.0 & 0.0 & -6.8 & 20.0 & -11.3 & 0.0 & 0.0 & -0.5 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
\hline
-1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\
0.0 & -2.1 & 0.0 & 0.0 & 0.0 & -2.1 & 0.0 & 0.0 & -4.5 & 23.1 & -13.5 & 0.0 \\
0.0 & 0.0 & -0.5 & 0.0 & 0.0 & 0.0 & -0.5 & 0.0 & 0.0 & -6.8 & 20.0 & -11.3 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0
\end{array}\right)$$

となる。

これを j 方向番号付けに変えて、

$$\left(\begin{array}{ccc|ccc|ccc|ccc}
37.0 & 0.0 & 0.0 & -12.0 & -12.0 & -12.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
-1.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
-1.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
\hline
-4.5 & 0.0 & 0.0 & 23.1 & -2.1 & -2.1 & -13.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & -4.5 & 0.0 & -2.1 & 23.1 & -2.1 & 0.0 & -13.5 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & -4.5 & -2.1 & -2.1 & 23.1 & 0.0 & 0.0 & -13.5 & 0.0 & 0.0 & 0.0 \\
\hline
0.0 & 0.0 & 0.0 & -6.8 & 0.0 & 0.0 & 20.0 & -0.5 & -0.5 & -11.3 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & -6.8 & 0.0 & -0.5 & 20.0 & -0.5 & 0.0 & -11.3 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -6.8 & -0.5 & -0.5 & 20.0 & 0.0 & 0.0 & -11.3 \\
\hline
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0
\end{array}\right)$$

となる。ここから LU 分解の計算範囲を対角線の成分から $2N_\varphi + 1 = 5$ 離れた所までとする。

次に Dirichlet 境界条件と原点のコピー $(U^{n+1}_{0,j} = U^{n+1}_{0,0})$ の行を取り除く操作を加えると、

$$\left(\begin{array}{c|ccc|ccc}
37.0 & -12.0 & -12.0 & -12.0 & 0.0 & 0.0 & 0.0 \\
\hline
-4.5 & 23.1 & -2.1 & -2.1 & -13.5 & 0.0 & 0.0 \\
-4.5 & -2.1 & 23.1 & -2.1 & 0.0 & -13.5 & 0.0 \\
-4.5 & -2.1 & -2.1 & 23.1 & 0.0 & 0.0 & -13.5 \\
\hline
0.0 & -6.8 & 0.0 & 0.0 & 20.0 & -0.5 & -0.5 \\
0.0 & 0.0 & -6.8 & 0.0 & -0.5 & 20.0 & -0.5 \\
0.0 & 0.0 & 0.0 & -6.8 & -0.5 & -0.5 & 20.0
\end{array}\right)$$

となり、計算範囲を対角成分から $N_\varphi = 3$ として計算量を大幅に減らせることがわかる。

これまでのことを整理したプログラムは次のようになる。

```
/*
 * heat2d-i-en3.c ---円盤状の 2 次元熱方程式を陰解法で解く
 *          j 方向番号付け 効率を考えたプログラム
 *   コンパイルするには
 *     gcc -c bandlu.c
 *     gcc -c call_gnuplot.c
 *
 *     ccmg heat2d-i-en3.c bandlu.c call_gnuplot.c
 */
#include <stdio.h>
#include <math.h>
#include <matrix.h>
#include "bandlu.h"
#include "call_gnuplot.h"
#define  psi(i,j) ((i-1) * mm + j+1)

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

int main()
{
 double ri ,ri2, phi_j;
 int N_r, N_p, mm, NN, i, j, p, q, n, skip, nMax, L;
 matrix Uk, A;
 double *B, *vector_U;
 double h_r, h_p, lambda_r, lambda_p, lambda, tau, t, Tmax, dt, M, ex;
 double theta;
 char label[200];

  /* πの値*/
  pi = 4.0 * atan(1.0);

  /* 区間の分割数 */
   printf("Nr, Ntheta: "); scanf("%d %d", &N_r, &N_p);

  mm = N_p;
  NN = (N_r-1)*N_p + 1;
```

```
/* 空間の刻み幅 */
h_r = 1.0 / N_r;
h_p = 2.0 * pi / N_p;


/* 行列、ベクトルを記憶する変数のメモリー割り当て */
 if ((Uk = new_matrix(N_r+1, N_p+1)) == NULL) {
    fprintf(stderr, "数列 U^k を記憶する領域の確保に失敗\n");
    exit(1);
 }
 if ((A = new_matrix(NN, NN)) == NULL) {
    fprintf(stderr, "係数行列 A を記憶する領域の確保に失敗\n");
    exit(1);
 }
 if ((B = (double *) malloc(sizeof(double) * NN)) == NULL) {
    fprintf(stderr, "B を記憶する領域の確保に失敗\n");
    exit(1);
 }

/* θ法の重みの決定 */
 printf("θ (0≦θ≦1) :  "); scanf("%lf", &theta);

 if (theta == 1.0) {
     printf("τ = "); scanf("%lf", &tau);
 } else {
   printf("τ (≦%g         ≡安定性条件): ",
          (h_r*h_r*h_p*h_p) / (2.0 * (1-theta)*(1+h_p*h_p)));
     scanf("%lf", &tau);
  }

  /* λr, λθ */
  lambda_r = tau / (h_r * h_r);
  lambda_p = tau / (h_p * h_p);
  /* 結果を出力する時間間隔の決定 */
  printf("Tmax: "); scanf("%lf", &Tmax);
  printf("Δt(>=%g): ", tau);  scanf("%lf", &dt);
  if (dt < tau) {
     dt = tau;
```

```
    }
    skip = rint(dt / tau);

    /* GNUPLOT の準備 */
    open_gnuplot();

  /* 初期値の設定 */
    for (i = 0; i <= N_r; i++) {
      ri = (i) * h_r;
      for (j = 0; j <= N_p; j++)
        Uk[i][j] = u0(ri, (j) * h_p);
    }

    disk(N_r, N_p, Uk, "t=0");

    /* 係数行列の値のセット */
    /* 0 クリア */
    for(p = 0; p < NN; p++){
      for(q = 0; q < NN; q++)
A[p][q] = 0.0;
      }
    /* 原点の係数 */
    A[0][0] = 1.0 + 4.0*theta*lambda_r;

    for(j = 0; j <= N_p-1; j++){
      L =  psi(1,j);
      A[0][L] = - (4.0*theta*lambda_r) / N_p;
      }


 /* 内点での係数 */
    for(i = 1; i< N_r; i++){
         ri = (i)*h_r;
         ri2 = ri*ri;
 for(j = 0; j <= N_p-1; j++){
  L = psi(i,j);
  int L1 = L-1;
  int L2 = L+1;
  int Lm = L-mm;
```

```
  if(j == 0)
    L1 = psi(i,N_p-1);

          if(j == N_p-1)
             L2 = psi(i,0);

 if(i == 1)
    Lm = 0;

 if(i != N_r-1)
   A[L][L+mm] = - theta* lambda_r* (1.0+h_r/(2.0*ri));

A[L][Lm] = - theta* lambda_r * (1.0 - h_r/(2.0*ri));
A[L][L] = 1.0 +2.0*theta*lambda_r +(2.0*theta*lambda_p)/ri2;
A[L][L1] = - theta * lambda_p / ri2;
A[L][L2] = - theta * lambda_p / ri2;
}
    }


   /* 係数行列 LU 分解 */
bandlu(A, NN, mm);

  /* 境界条件 */
   for(j = 0; j <= N_p; j++){
     Uk[N_r][j] = 0.0;
   }

   /* 時間に関するループ */
   nMax = rint(Tmax / tau);
   for (n = 1; n <= nMax; n++) {


     /* 連立 1 次方程式の右辺を用意する */
     /* 内部の格子点 */
     for (i = 1; i < N_r; i++){
     ri = (i)*h_r;
     ri2 = ri*ri;
     for(j = 0; j<= N_p-1; j++){
```

```
 L = psi(i,j);
 int jm1 = j-1;
 int jm2 = j+1;
     if(jm1 == -1)
   jm1 = N_p-1;
     if(jm2 == N_p)
           jm2 = 0;
       B[L]= (1.0-2.0*(1.0-theta)*lambda_r
           -(2.0*(1.0-theta)*lambda_p)/ri2)*Uk[i][j]
           +(1.0-theta)*lambda_r*((1.0+h_r/(2.0*ri))*Uk[i+1][j]
           +(1.0-h_r/(2.0*ri))*Uk[i-1][j])
           +((1.0-theta)*lambda_p/ri2)*(Uk[i][jm1]+Uk[i][jm2]);
     }
     }

     /* 原点 */
     B[0] = (1.0 - 4.0*(1.0 - theta)*lambda_r)*Uk[0][0];
   for(j = 0; j <= N_p-1; j++){
     B[0] += (4.0*(1.0 - theta)*lambda_r / N_p)*Uk[1][j];
   }

/* A  vector_U = B を解く */
   bandsolve(A, B, NN, mm);

 /* 更新 */
   Uk[0][0] = B[0];
   for(j = 1; j<= N_p; j++)
     Uk[0][j] = Uk[0][0];

   for(i = 1; i< N_r; i++){
     for(j = 0; j <= N_p-1; j++){
       L = psi(i,j);
       Uk[i][j] = B[L];
     }
     Uk[i][N_p]= B[psi(i,0)];
   }

 t = n * tau;
```

```
 /* Δ t の整数倍の時刻ではグラフをを描く */
   if (n % skip == 0){
     sprintf(label, "t=%g", t);
     disk(N_r, N_p, Uk, label);
   }

 /* 誤差を測る */
   M = 0.0;
   for(i = 0; i <= N_r; i++){
     ri = (i)*h_r;
     for(j = 0; j <= N_p; j++){
       double e;
       phi_j = (j)*h_p;
       e = fabs(exactu(ri, phi_j, t) - Uk[i][j]);
       if(e > M)
 M = e;
     }
   }
   printf("n=%d, norm=%g, t=%g, 誤差=%g\n",
  n, maxnorm(N_r, N_p, Uk), t, M);

}
  close_gnuplot();

 return 0;
}



 /* 厳密解を計算する関数 */

#define mu01    (2.404825557695771998)
#define mu11    (3.831705970207510692)

     /* 初期データ */
double u0(double r, double phi)
  {
    return j0(mu01* r)+ j1(mu11* r)* (cos(phi)+ sin(phi));
  }
```

```
    /* 厳密解 */
double exactu(double r, double phi, double t)
  {
    return exp(- mu01* mu01* t)* j0(mu01* r)
          + exp(- mu11* mu11* t)*j1(mu11* r)*(cos(phi) + sin(phi));
  }

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

1.5 実験結果

実際に数値計算をした結果を以下に記す。
プログラム中では厳密解を、

$$u(r,\varphi,t) = e^{-{\mu_{01}}^2 t} J_0(\mu_{01} r) + e(-{\mu_{11}}^2 t) J_1(\mu_{11} r)(\cos\varphi + \sin\varphi). \tag{1.24}$$

初期値は、

$$f(r,\varphi) J_0(\mu 01 r) + J_1(\mu_{11} r)(\cos\varphi + \sin\varphi). \tag{1.25}$$

1.5.1 差分解の精度

最大ノルムは絶対値 $|U_{i,j}^n|$ の最大のもの、誤差は厳密解と最大ノルムの差を絶対値でとっている。次の表は分割数を 2 倍、τ を $\frac{1}{4}$ 倍にしていき、$t=1$ のときの数値を表している。

陽解法 ($\theta=0$)

N_r	N_φ	τ	最大ノルム	誤差	上の誤差との比
5	20	0.00008	0.00349125	0.000412356	
10	40	0.00002	0.00317918	0.000100291	0.243214601
20	80	0.000005	0.00310379	0.0000249018	0.248229546

陰解法 ($\theta=1$)

N_r	N_φ	τ	最大ノルム	誤差	上の誤差との比
5	20	0.00008	0.00350019	0.000421301	
10	40	0.00002	0.00318129	0.000102395	0.243044795
20	80	0.000005	0.00310431	0.0000254194	0.24824845

分割数を 2 倍、τ を $\frac{1}{4}$ 倍と細かくするにつれて誤差は約 $\frac{1}{4}$ 倍になり、厳密解に近い結果得られていく。

1.5.2 安定性

陽解法の安定条件は、

$$\min_{1\le i\le N_r-1}(1-2\lambda_r-\frac{2\lambda_\varphi}{r_i^2})\ge 0. \tag{1.26}$$

と予想されている (遠藤, 高木, 内藤 [6])。

$$\begin{aligned}
&1-\frac{2\tau}{h_r^2}-\frac{2\tau}{h_\varphi^2 h_r^2}\ge 0\\
\Leftrightarrow\ &\frac{2\tau}{h_r^2}+\frac{2\tau}{h_\varphi^2 h_r^2}\le 1\\
\Leftrightarrow\ &\tau(\frac{2}{h_r^2}+\frac{2}{h_\varphi^2 h_r^2})\le 1\\
\Leftrightarrow\ &\tau\le\frac{1}{\frac{2}{h_r^2}+\frac{2}{h_\varphi^2 h_r^2}}\\
\Leftrightarrow\ &\tau\le\frac{h_\varphi^2 h_r^2}{2(1+h_\varphi^2)}.
\end{aligned} \tag{1.27}$$

このように τ を制限すれば、安定性は成立するようだが、分割を細かくとき τ を非常に小さくする必要があり満足できるものではない。

一方、θ 法では $0\le\theta<1$ であるとき安定性の条件は、当初陽解法の場合からの類推で

$$\min_{1\le i\le N_r-1}(1-2(1-\theta)\lambda_r-\frac{2(1-\theta)\lambda_\varphi}{r_i^2})\ge 0 \tag{1.28}$$

であると予想していた。これは、

$$
\begin{aligned}
&1-\frac{2(1-\theta)\tau}{h_r^2}-\frac{2(1-\theta)\tau}{h_\varphi^2 h_r^2}\geq 0\\
\Leftrightarrow\ &\frac{2(1-\theta)\tau}{h_r^2}+\frac{2(1-\theta)\tau}{h_\varphi^2 h_r^2}\leq 1\\
\Leftrightarrow\ &\tau(\frac{2(1-\theta)}{h_r^2}+\frac{2(1-\theta)}{h_\varphi^2 h_r^2})\leq 1\\
\Leftrightarrow\ &\tau\leq\frac{1}{\frac{2(1-\theta)}{h_r^2}+\frac{2(1-\theta)}{h_\varphi^2 h_r^2}}\\
\Leftrightarrow\ &\tau\leq\frac{h_\varphi^2 h_r^2}{2(1-\theta)(1+h_\varphi^2)} \qquad (1.29)
\end{aligned}
$$

と書き直せる。

上の安定条件で数値実験を検証し $N_r=20, N_\varphi=80, \tau=0.001$ として θ の値を変えてみると、

		$t=1$		$t=2$	
θ	予想された安定条件	誤差	安定 or 不安定	誤差	安定 or 不安定
0.48	$\tau\leq 1.47372\times 10^{-5}$	2.50299×10^{12}	不安定	2.56835×10^{40}	不安定
0.49	$\tau\leq 1.50262\times 10^{-5}$	3.74873×10^{-5}	安定	$4.60679\ \times10^{5}$	不安定
0.50	$\tau\leq 1.53267\times 10^{-5}$	2.51108×10^{-5}	安定	1.5547×10^{-7}	安定
0.51	$\tau\leq 1.50262\times 10^{-5}$	2.61461×10^{-5}	安定	1.61898×10^{-7}	安定

ここから上の安定条件 (1.29) は成り立たないと考えられ、$\theta\geq 0.5$ では無条件安定する可能性が出てきた。

陰解法の $\theta=1$ の場合には無条件に安定すると考えられる。

次の数値結果の表を見てほしい。

$N_r=10, N_\varphi=40$(陽解法の安定条件：$\tau\leq 1.20399\times10^{-4}$) として $Tmax=1$ まで調べた時

	陽解法		陰解法	
τ	誤差	安定 or 不安定	誤差	安定 or 不安定
1.2×10^{-4}	9.5054×10^{-5}	安定	1.07677×10^{-4}	安定
1.20399×10^{-4}	9.49995×10^{-5}	安定	1.0766×10^{-4}	安定
1.23×10^{-4}	1.00966(t=0.249567)	不安定	1.0782×10^{-4}	安定

となる。

$\theta=0$ の場合は、τ を安定条件より大きくとると不安定な値 (誤差が大きい) が出てくるが、$\theta=1$ の場合は、安定した値が出てくる。

N_r, N_φ をそれぞれ 2 倍し、差分方程式のバランスを保つため τ を $\frac{1}{4}$ 倍して考える。

$N_r=20, N_\varphi=80$（陽解法の安定条件：$\tau\leq 7.6636e-6$）として、$Tmax=1$ まで調べた時、

	陽解法		陰解法	
τ	誤差	安定 or 不安定	誤差	安定 or 不安定
3×10^{-5}	3.7951(t=6.3×10^{-4})	不安定	2.67149×10^{-5}	安定
3.009775×10^{-5}	4.41213(t=6.32095×10^{-4})	不安定	2.67183×10^{-5}	安定
4 3.075×10^{-5}	6.13215(t=6.4575×10^{-4})	不安定	2.67538×10^{-5}	安定

陽解法は安定条件が厳しいため不安定な値が得られるが、陰解法では $N_r = 10, N_\varphi = 40$ の時と比べ、誤差が $\frac{1}{4}$ 倍になって安定した値が得られる。

しかし、無条件安定しても τ を大きくしたときに誤差が大きくなり信頼性がなくなる。そのため, この場合 $\tau = O(h_r^2 + h_\varphi^2)$ とすればよいだろう。

厳密解が正確な値がとれていないのではっきりとしたことがいえないが、もし $\tau = 0.5$ 以上で無条件安定するならば、$\tau = 1$ よりも $\tau = 0.5$ のほうが誤差は小さい値がでる。

1.5.3 不安定の検証

安定条件を越えて τ に値を入れても安定した結果が得られる場合がある。ここで、どのような条件がそろった場合に、不安定な値が出てくるか疑問となった。まず、陽解法の場合、安定性の条件は

$$\min_{1 \leq i \leq N_r - 1} (1 - 2(1-\theta)\lambda_r - \frac{2(1-\theta)\lambda_\varphi}{r_i^2}) \geq 0$$

であると予測している (遠藤, 高木, 内藤 [6])。

上の条件が成り立たないと差分解が爆発的に増大する場合がある。観察の結果、上の条件が成り立たない場合に $U_{i,j}^n$ の値が負になった時、振動を起こすことに注目した。

$$\min_{1 \leq i \leq N_r - 1} (1 - 2(1-\theta)\lambda_r - \frac{2(1-\theta)\lambda_\varphi}{r_i^2}) < 0$$

上の式では $i = 1$ の時最も小さい値が取れるので、振動が始まるのは i = 1 のときの $\min_{1 \leq i \leq N_r - 1}(U_{1,j}^n)$ の値が負になったときである。

つまり、不安定になるのは

$$\tau > \frac{h_\varphi^2 h_r^2}{2(1 + h_\varphi^2)}$$

$$\min(U_{1,j}^n) < 0$$

の条件が揃う時、不安定になる。　が成り立たなければ安定し続けると予想した。

そこで、プログラムで $\min(U_{1,j}^n)$ を表すように書き加えて、上の表と同じ条件で調べる。

$N_r = 10, N_\varphi = 40$(陽解法の安定条件：$\tau \leq 1.20399 \times 10^{-4}$) の場合

τ=1.2×10^{-4}	陽解法		陰解法	
t	$\min(U_{1,j}^{n+1})$	誤差	$\min(U_{1,j}^{n+1})$	誤差
0.99948	0.00313754	9.52716×10^{-5}	0.00315001	1.07923×10^{-4}
0.9996	0.00313538	9.52172×10^{-5}	0.00314784	1.07862×10^{-4}
0.99972	0.00313322	9.51628×10^{-5}	0.00314567	1.078×10^{-4}
0.99984	0.00313105	9.51084×10^{-5}	0.0031435	1.07739×10^{-4}
0.99996	0.00312889	9.5054×10^{-5}	0.00314133	1.07677×10^{-4}
τ=1.20399×10^{-4}	陽解法		陰解法	
t	$\min(U_{1,j}^{n+1})$	誤差	$\min(U_{1,j}^{n+1})$	誤差
0.999552	0.00313622	9.52177×10^{-5}	0.00314872	1.07907×10^{-4}
0.999673	0.00313404	9.51631×10^{-5}	0.00314654	1.07845×10^{-4}
0.999793	0.00313187	9.51086×10^{-5}	0.00314436	1.07784×10^{-4}
0.999914	0.00312971	9.5054×10^{-5}	0.00314219	1.07722×10^{-4}
1.00003	0.00312754	9.49995×10^{-5}	0.00314001	1.0766×10^{-4}
τ=1.23×10^{-4}	陽解法		陰解法	
t	$\min(U_{1,j}^{n+1})$	誤差	$\min(U_{1,j}^{n+1})$	誤差
0.239973	0.0026766	0.239167	0.240062	2.34888×10^{-3}
0.240096	-0.00202304	0.243606	0.239901	2.34787×10^{-3}
0.240219	-0.00661054	0.248131	0.23974	2.34686×10^{-3}
⋮	⋮	⋮	⋮	⋮
0.999744	-1.30626×10^{49}	1.30626×10^{49}	0.00314539	1.07946×10^{-4}
0.999867	-1.3307×10^{49}	1.3307×10^{49}	0.00314317	1.07883×10^{-4}
0.99999	-1.3556×10^{49}	1.3556×10^{49}	0.00314095	1.0782×10^{-4}

$tau = 1.2$,$tau = 1.20399$ の場合、安定条件 $\tau \leq 1.20399 \times 10^{-4}$ を満たしているため安定する。

τ=1.23×10^{-4} の場合、安定条件が成り立っていないのと不安定になる前に $\min(U_{1,j}^{n+1})$ が負になり始めた。

$N_r = 20, N_\varphi = 80, \tau = 3 \times 10^{-5}$(陽解法の安定条件：$\tau \leq 7.6636e-6$) の場合だとわかりやすいであろう。

τ=3×10^{-5}	陽解法		陰解法	
t	$\min(U_{1,j}^{n+1})$	誤差	$\min(U_{1,j}^{n+1})$	誤差
5.1×10^{-4}	0.859474	0.00200941	0.8597	9.01101×10^{-5}
5.4×10^{-4}	0.853891	0.0133156	0.859591	9.43433×10^{-5}
5.7×10^{-4}	0.782585	0.0869772	0.859481	9.84814×10^{-5}
6×10^{-4}	0.321002	0.574231	0.859372	1.02527×10^{-4}
6.3×10^{-4}	-2.92218	3.79151	0.859263	1.06484×10^{-4}
6.6×10^{-4}	-23.3165	25.0756	0.859153	1.10354×10^{-4}
6.9×10^{-4}	-165.182	166.051	0.859043	1.14141×10^{-4}
7.2×10^{-4}	-1061.7	1100.93	0.858934	1.17847×10^{-4}

陽解法で、安定条件 $\tau \leq 7.6636e-6$ は成り立っていない。次に、$\min(U_{1,j}^{n+1})$ が負になり始めて

から誤差の増え方が大きくなった。

この結果から、

$$\tau > \frac{h_\varphi^2 h_r^2}{2(1+h_\varphi^2)}$$

$$\min(U_{1,j}^n) < 0$$

が揃った時点で不安定になる。

安定した値出し続けるなら、　の条件が成り立たないようにすればよいので

$$\tau \leq \frac{h_\varphi^2 h_r^2}{2(1+h_\varphi^2)}$$

であることが安定条件になるだろう。

第2章　Swartztrauber-Sweet近似

今まで、Laplacian の極座標表示

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

をそのまま近似して考えてきたが、一般的には Swartztrauber-Sweet 近似の方が主流である。

2.1　Swartztrauber-Sweet近似の導出

極座標変換した Laplacian は、

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

であったが、

$$\triangle = \frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial}{\partial r}) + \frac{1}{r^2}\frac{\partial^2}{\partial \varphi^2} \tag{2.1}$$

と書き直すことができる。

$N_r, N_\varphi \in \boldsymbol{N}$ に対して、

$$h_r := \frac{1}{N_r}, \quad h_\varphi := \frac{2\pi}{N_\varphi},$$

$$r_i := ih_r \quad (i = 0, \frac{1}{2}, 1, \frac{3}{2} \ldots, \frac{2N_r - 1}{2}, N_r),$$

$$\varphi_j := jh_\varphi \quad (j = 0, 1, \ldots, N_\varphi).$$

$\tau > 0$ を固定して

$$t_n := n\tau \quad (n = 0, 1, 2, \ldots).$$

$$\lambda_r := \frac{\tau}{h_r^2}, \ \lambda_\varphi := \frac{\tau}{h_\varphi^2}.$$

と定義する。

$$u_{i,j} := u(r_i, \varphi_j) \quad (i = 0, 1, \ldots, N_r; \ j = 0, 1, \ldots, N_\varphi).$$

として、

関係式

$$\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial u_{i,j}}{\partial r})_{r=r_i} = \frac{1}{r}\frac{r_{i+\frac{1}{2}}\frac{u_{i+1,j}-u_{i,j}}{h_r} - r_{i-\frac{1}{2}}\frac{u_{i,j}-u_{i-1,j}}{h_r}}{h_r} + O(h_r^2)$$

$$= \frac{1}{r_i h_r^2}(r_{i+\frac{1}{2}}(u_{i+1,j} - u_{i,j}) - r_{i-\frac{1}{2}}(u_{i,j} - u_{i-1,j})) \tag{2.2}$$

を用いて差分近似をすれば、

$$\triangle u(r_i, \varphi_j) = \frac{1}{r_i h_r^2}(r_{i+\frac{1}{2}}(u_{i+1,j} - u_{i,j}) - r_{i-\frac{1}{2}}(u_{i,j} - u_{i-1,j}))$$

$$+\frac{1}{r_i^2 h_\varphi^2}(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}) + O(h_r^2 + h_\varphi^2) \tag{2.3}$$

とできる。

この近似を Swartztrauber-Sweet 近似という (山本 [3])。

これを以下 SS 近似と略する。

2.2 Swartztrauber-Sweet 近似による θ 法

2.2.1 原点以外の点での差分方程式

原点以外の点では、まず (2.3) を用いて陽解法による差分方程式、陰解法の差分方程式を出すと、

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

$$+\frac{1}{r_i^2 h_\varphi^2}(U_{i,j+1}^n - 2U_{i,j}^n + U_{i,j-1}^n), \tag{2.4}$$

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

$$+\frac{1}{r_i^2 h_\varphi^2}(U_{i,j+1}^n - 2U_{i,j}^n + U_{i,j-1}^n) \tag{2.5}$$

となる。

上の θ 法と同様に、この差分方程式を $1-\theta$ と θ の重みをつけて重ね合わせると、

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

$$+\frac{1}{r_i^2 h_\varphi^2}(U_{i,j+1}^n - 2U_{i,j}^n + U_{i,j+1}^n)]$$

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

$$+\frac{1}{r_i^2 h_\varphi^2}(U_{i,j+1}^{n+1} - 2U_{i,j}^{n+1} + U_{i,j+1}^{n+1})] \tag{2.6}$$

$$(i = 1, 2, \ldots, N_r - 1; j = 0, 1, \ldots, N_\varphi - 1; n = 0, 1, \ldots).$$

分母を払って整理すると、

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

$$-\frac{\theta\lambda_\varphi}{r_i^2}(U^{n+1}_{i,j+1}+U^{n+1}_{i,j-1})=$$

$$(1-\frac{r_{i+\frac{1}{2}}(1-\theta)\lambda_r}{r_i}-\frac{r_{i-\frac{1}{2}}(1-\theta)\lambda_r}{r_i}-\frac{2(1-\theta)\lambda_\varphi}{r_i^2})U^n_{i,j}$$

$$+\frac{(1-\theta)\lambda_r}{r_i}(r_{i+\frac{1}{2}}U^n_{i+1,j}+r_{i-\frac{1}{2}}U^n_{i-1,j})+\frac{(1-\theta)\lambda_\varphi}{r_i^2}(U^n_{i,j+1}+U^n_{i,j-1}) \tag{2.7}$$

$$(i=1,2,\ldots,N_r-1;j=0,1,\ldots,N_\varphi-1;n=0,1,\ldots).$$

通常の差分近似の θ 法で出した差分方程式と比べてみると、係数が若干変化しただけでそれほどの違いはないことがわかる。

2.2.2 原点での差分方程式

原点においてはこれまで通り、

$$(1+4\theta\lambda_r)U^{n+1}_{0,0}-\frac{4\theta\lambda_r}{N_\varphi}\sum_{j=0}^{N_\varphi-1}U^{n+1}_{1,j}=(1-4(1-\theta)\lambda_r)U^n_{0,0}+\frac{4(1-\theta)\lambda_r}{N_\varphi}\sum_{j=0}^{N_\varphi-1}U^n_{1,j} \tag{2.8}$$

$$(j=0,1,\ldots,N_\varphi-1;n=0,1,\ldots)$$

$$U^n_{0,j}=U^n_{0,0},U^{n+1}_{0,j}=U^{n+1}_{0,0} \qquad (j=0,1,\ldots,N_\varphi-1)$$

を採用する。

SS 近似のプログラムでは、上の θ 法のプログラムを元に作れば良い。

2.2.3 Swartztrauber-Sweet 近似を用いたプログラム

```
/*
 * ss.c ---円盤状の 2 次元熱方程式を ss 近似で解く
 *          j 方向番号付け 効率を考えたプログラム
 *   コンパイルするには
 *     gcc -c bandlu.c
 *     gcc -c call_gnuplot.c
 *
 *     ccmg ss.c bandlu.o call_gnuplot.o
 */
#include <stdio.h>
#include <math.h>
#include <matrix.h>
#include "bandlu.h"
#include "call_gnuplot.h"
```

```
#define  psi(i,j) ((i-1) * mm + j+1)

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

int main()
{
  double ri ,ri2, ri3, ri4, phi_j;
  int N_r, N_p, mm, NN, i, j, p, q, n, skip, nMax, L;
  matrix Uk, A;
  double *B, *vector_U;
  double h_r, h_p, lambda_r, lambda_p, lambda, tau, t, Tmax, dt, M, ex;
  double theta;
  char label[200];

  /* πの値*/
  pi = 4.0 * atan(1.0);

  /* 区間の分割数 */
   printf("Nr, Ntheta: "); scanf("%d %d", &N_r, &N_p);

  mm = N_p;
  NN = (N_r-1)*N_p + 1;

  /* 空間の刻み幅 */
  h_r = 1.0 / N_r;
  h_p = (2.0 * pi) / N_p;


  /* 行列、ベクトルを記憶する変数のメモリー割り当て */
    if ((Uk = new_matrix(N_r+1, N_p+1)) == NULL) {
        fprintf(stderr, "数列 U^k を記憶する領域の確保に失敗\n");
        exit(1);
    }
    if ((A = new_matrix(NN, NN)) == NULL) {
        fprintf(stderr, "係数行列 A を記憶する領域の確保に失敗\n");
        exit(1);
```

```
  }
  if ((B = (double *) malloc(sizeof(double) * NN)) == NULL) {
      fprintf(stderr, "B を記憶する領域の確保に失敗\n");
      exit(1);
  }

  /* θ法の重みの決定 */
  printf("θ (0≦θ≦1) :  "); scanf("%lf", &theta);

  if (theta == 1.0) {
      printf("τ = "); scanf("%lf", &tau);
  } else {
    printf("τ (≦%g         ≡安定性条件): ",
           (h_r * h_r * h_p * h_p) / (2* (1 - theta) *(1 + h_p * h_p)));
     scanf("%lf", &tau);
  }

  /* λ r, λ θ */
  lambda_r = tau / (h_r * h_r);
  lambda_p = tau / (h_p * h_p);
  /* 結果を出力する時間間隔の決定 */
  printf("Tmax: "); scanf("%lf", &Tmax);
  printf("Δ t(>=%g): ", tau);  scanf("%lf", &dt);
  if (dt < tau) {
     dt = tau;
  }
  skip = rint(dt / tau);

  /* GNUPLOT の準備 */
  open_gnuplot();

/* 初期値の設定 */
  for (i = 0; i <= N_r; i++) {
    ri = (i) * h_r;
    for (j = 0; j <= N_p; j++)
      Uk[i][j] = u0(ri, (j) * h_p);
  }

  disk(N_r, N_p, Uk, "t=0");
```

```
   /* 係数行列の値のセット */
   /* 0 クリア */
   for(p = 0; p < NN; p++){
     for(q = 0; q < NN; q++)
       A[p][q] = 0.0;
     }
   /* 原点の係数 */
   A[0][0] = 1.0 + 4.0*theta*lambda_r;

   for(j = 0; j <= N_p-1; j++){
     L =  psi(1,j);
     A[0][L] = - (4.0*theta*lambda_r) / N_p;
     }


/* 内点での係数 */
   for(i = 1; i< N_r; i++){
        ri = (i)*h_r;
        ri2 = ri*ri;
        ri3 = (i+(1.0/2.0))*h_r;
        ri4 = (i-(1.0/2.0))*h_r;
        for(j = 0; j <= N_p-1; j++){
         L = psi(i,j);
         int L1 = L-1;
         int L2 = L+1;
         int Lm = L-mm;
         if(j == 0)
           L1 = psi(i,N_p-1);

         if(j == N_p-1)
           L2 = psi(i,0);

         if(i == 1)
           Lm = 0;

         if(i != N_r-1)
           A[L][L+mm] =  (- ri3 * theta * lambda_r)/ri;
```

```
        A[L][Lm] =  (- ri4 * theta * lambda_r)/ri;
        A[L][L] = 1.0 + (ri3*theta*lambda_r)/ri
                  + (ri4*theta*lambda_r)/ri + (2.0*theta*lambda_p)/ri2;
        A[L][L1] = - theta * lambda_p / ri2;
        A[L][L2] = - theta * lambda_p / ri2;
       }
  }


 /* 係数行列 LU 分解 */
      bandlu(A, NN, mm);

/* 境界条件 */
 for(j = 0; j <= N_p; j++){
   Uk[N_r][j] = 0.0;
 }

 /* 時間に関するループ */
 nMax = rint(Tmax / tau);
 for (n = 1; n <= nMax; n++) {


   /* 連立 1 次方程式の右辺を用意する */
   /* 内部の格子点 */
   for (i = 1; i < N_r; i++){
   ri = (i)*h_r;
   ri2 = ri*ri;
   ri3 = (i+(1.0/2.0))*h_r;
   ri4 = (i-(1.0/2.0))*h_r;
   for(j = 0; j<= N_p-1; j++){
       L = psi(i,j);
       int jm1 = j-1;
       int jm2 = j+1;
   if(jm1 == -1)
         jm1 = N_p-1;
   if(jm2 == N_p)
         jm2 = 0;
   B[L] =  ( 1.0 - (ri3*(1-theta)*lambda_r)/ri - (ri4*(1-theta)*lambda_r)/ri
         - (2.0*(1-theta)*lambda_p)/ri2 )*Uk[i][j]
```

```
            + ((1.0-theta)*lambda_r/ri)*(ri3*Uk[i+1][j] + ri4*Uk[i-1][j])
            + ((1.0 - theta)*lambda_p / ri2)*(Uk[i][jm1] + Uk[i][jm2]);
      }
      }

      /* 原点 */
      B[0] = (1.0 - 4.0*(1.0 - theta)*lambda_r)*Uk[0][0];
    for(j = 0; j <= N_p-1; j++){
      B[0] += (4.0*(1.0 - theta)*lambda_r / N_p)*Uk[1][j];
    }

/* A  vector_U = B を解く */
    bandsolve(A, B, NN, mm);

 /* 更新 */
    Uk[0][0] = B[0];
    for(j = 1; j<= N_p; j++)
      Uk[0][j] = Uk[0][0];

    for(i = 1; i< N_r; i++){
      for(j = 0; j <= N_p-1; j++){
        L = psi(i,j);
        Uk[i][j] = B[L];
      }
      Uk[i][N_p]= B[psi(i,0)];
    }

 t = n * tau;

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

 /* 誤差を測る */
    M = 0.0;
    for(i = 0; i <= N_r; i++){
      ri = (i)*h_r;
```

```
      for(j = 0; j <= N_p; j++){
        double e;
        phi_j = (j)*h_p;
        e = fabs(exactu(ri, phi_j, t) - Uk[i][j]);
        if(e > M)
          M = e;
      }
    }
    printf("n=%d, norm=%g, t=%g, 誤差=%g\n",
           n, maxnorm(N_r, N_p, Uk), t, M);

}
  close_gnuplot();

 return 0;
}



 /* 厳密解を計算する関数 */

#define mu01    (2.404825557695771998)
#define mu11    (3.831705970207510692)

     /* 初期データ */
double u0(double r, double phi)
  {
    return j0(mu01 * r) + j1(mu11 * r) * (cos(phi) + sin(phi));
  }

    /* 厳密解 */
double exactu(double r, double phi, double t)
          {
            return exp(- mu01* mu01* t)* j0(mu01* r)
                 + exp(- mu11* mu11* t)* j1(mu11* r)* (cos(phi)+sin(phi));
          }

double maxnorm(int m, int n, matrix Uk)
      {
```

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

2.3 実験結果

2.3.1 差分解の精度

上の θ 法と同じ条件で表を作る。

陽解法 ($\theta = 0$)

N_r	N_φ	τ	最大ノルム	誤差	上の誤差との比
5	20	0.00008	0.00349125	0.000412356	
10	40	0.00002	0.00317918	0.000100291	0.243214601
20	80	0.000005	0.00310379	0.0000249018	0.248229546

陰解法 ($\theta = 1$)

N_r	N_φ	τ	最大ノルム	誤差	上の誤差との比
5	20	0.00008	0.00350019	0.000421301	
10	40	0.00002	0.00318129	0.000102395	0.243044795
20	80	0.000005	0.00310431	0.0000254194	0.24824845

同様に、分割数を 2 倍,τ　を$\frac{1}{4}$ 倍と細かくするにつれて、誤差は約 $\frac{1}{4}$ 倍になり、厳密解に近い値が得られる。

これは、通常の Laplacian の差分近似の実験結果と比較して、「最大ノルム」「誤差」は同じ値をとっていることがわかる。このことは、2.3.3 で調べている。

2.3.2 安定性

SS 近似の差分方程式の安定条件は、通常の差分近似の差分方程式と同じように、

$$\min(1 - \frac{r_{i+\frac{1}{2}}(1-\theta)\lambda_r}{r_i} - \frac{r_{i-\frac{1}{2}}(1-\theta)\lambda_r}{r_i} - \frac{2(1-\theta)\lambda_\varphi}{r_i^2}) \geq 0$$

をとると、

$$\begin{aligned}
1 - \frac{r_{1+\frac{1}{2}}(1-\theta)\lambda_r}{r_1} - \frac{r_{1-\frac{1}{2}}(1-\theta)\lambda_r}{r_1} - \frac{2(1-\theta)\lambda_\varphi}{r_1^2} &\geq 0 \\
\frac{r_{1+\frac{1}{2}}(1-\theta)\lambda_r}{r_1} + \frac{r_{1-\frac{1}{2}}(1-\theta)\lambda_r}{r_1} + \frac{2(1-\theta)\lambda_\varphi}{r_1^2} &\leq 1 \\
\frac{\frac{3}{2}h_r(1-\theta)\frac{\tau}{h_r^2}}{h_r} + \frac{\frac{1}{2}h_r(1-\theta)\frac{\tau}{h_r^2}}{h_r} + \frac{2(1-\theta)\frac{\tau}{h_\varphi^2}}{h_r^2} &\leq 1 \\
\frac{3(1-\theta)\tau}{2h_r^2} + \frac{(1-\theta)\tau}{2h_r^2} + \frac{2(1-\theta)\tau}{h_r^2 h_\varphi^2} &\leq 1 \\
(1-\theta)\tau(\frac{2}{h_r^2} + \frac{2}{h_r^2 h_\varphi^2}) &\leq 1
\end{aligned}$$

$$\tau \leq \frac{h_r^2 h_\varphi^2}{2(1-\theta)(1+h_\varphi^2)} \tag{2.9}$$

となる。

これは、通常の差分方程式の安定条件と同じであるので、前節 1.4「安定性」を参考にしてもらいたい。

2.3.3 通常の差分近似との比較

それぞれの近似の最大ノルム、誤差について表をまとめた。

$N_r = 10, N_\varphi = 40$ の陽解法（安定条件：上から $\tau \leq 1.20399 \times 10^{-4}$,$\tau \leq 1.20399 \times 10^{-4}$,$\tau \leq 1.20399 \times 10^{-4}$）では、

τ=1.2×10^{-4}	通常の差分近似		SS 近似	
t	最大ノルム	誤差	最大ノルム	誤差
0.99948	0.00318344	9.52716×10^{-5}	0.00318344	9.52716×10^{-5}
0.9996	0.00318124	9.52172×10^{-5}	0.00318124	9.52172×10^{-5}
0.99972	0.00317904	9.51628×10^{-5}	0.00317904	9.51628×10^{-5}
0.99984	0.00317685	9.51084×10^{-5}	0.00317685	9.51084×10^{-5}
0.99996	0.00317466	9.5054×10^{-5}	0.00317466	9.5054×10^{-5}
τ=1.20399×10^{-4}	通常の差分近似		SS 近似	
t	最大ノルム	誤差	最大ノルム	誤差
0.999552	0.00318209	9.52177×10^{-5}	0.00318209	9.52177×10^{-5}
0.999673	0.00317988	9.51631×10^{-5}	0.00317988	9.51631×10^{-5}
0.999793	0.00317768	9.51086×10^{-5}	0.00317768	9.51086×10^{-5}
0.999914	0.00317548	9.5054×10^{-5}	0.00317548	9.5054×10^{-5}
1.00003	0.00317328	9.49995×10^{-5}	0.00317328	9.49995×10^{-5}
τ=1.23×10^{-4}	通常の差分近似		SS 近似	
t	最大ノルム	誤差	最大ノルム	誤差
0.239973	0.49304	0.239167	0.240062	2.34888×10^{-3}
0.240096	0.497194	0.243606	0.239901	2.34787×10^{-3}
0.240219	0.501626	0.248131	0.23974	2.34686×10^{-3}
⋮	⋮	⋮	⋮	⋮
0.999744	1.30626×10^{49}	1.30626×10^{49}	1.18574×10^{49}	1.1857×10^{49}
0.999867	1.3307×10^{49}	1.3307×10^{49}	1.20793×10^{49}	1.20793×10^{49}
0.99999	1.3556×10^{49}	1.3556×10^{49}	1.23054×10^{49}	1.23054×10^{49}

陽解法で調べたが安定しているときは、それぞれの最大ノルム, 誤差は同じであった。しかし、不安定になると最大ノルム, 誤差ともに同じではなかった。

不安定だが、誤差の大きさが SS 近似の方が若干小さいため、SS 近似のほうが極微小に精度が良いのではないだろうか？

第3章　あとがき

3.1　あとがき

ここまでが私の研究内容であるが、当初の目標よりも先に進むことができた。

しかしながら問題はある。安定条件において確実な証明ができていないこと、通常の差分近似とSS 近似の精度の違いがまだしっかりとわかっていないことが挙げられる。この 2 点がこれからの課題である。

さらに欲を言えば、プログラムにおいてはしっかりとできたと思うが、$N_r = 40, N_\varphi = 160\tau = 4 \times 10^{-7}$ を Tmax=1 まで計算するのに 3 時間以上かかる。そのため、これ以上に効率良くしたい。(どこを直せば良いかわからないが ...) また、今回は半径 1 の円盤で考えているが任意の半径 $\boldsymbol{R}$ において成り立つようなプログラムを作っておきたい。

3.2　その他のプログラム

[1]「効率を考えるプログラム」で使った係数行列の様子を見るプログラム
heat2d-i-disk.c heat2d-i-disk2.c heat2d-i-disk3.c
[2]「不安定になる条件」で使用した $\min(U^n_{1,j})$ を表すプログラム
jikken.c
[3] 帯行列を計算するプログラム
bandlu.c

関連図書

[1] 藤田 宏 「応用数学」 (放送大学教育振興会)

[2] 金子 晃 「偏微分方程式入門」 (東京大学出版会)

[3] 山本 哲郎 「数値解析入門 [増訂版]」 (サイエンス社)

[4] 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

[5] 松本 英久 「２次元領域における熱伝導方程式」
(1996 年度卒業研究レポート)
http://www.math.meiji.ac.jp/~mk/labo/report

[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)
http://www.math.meiji.ac.jp/~mk/labo/report

[7] 岡田 俊宣 「円盤、円柱領域における熱方程式に対する差分法」
(2004 年度卒業研究レポート)
http://www.math.meiji.ac.jp/~mk/labo/report