長方形領域における熱方程式に対する差分法

同次 Dirichlet 境界条件の場合, MATLAB を使って数値計算

桂田 祐史

2015 年 5 月 1 日, 2015 年 5 月 30 日, 2024 年 8 月 16 日 (組版し直し)

空間 1 次元の熱方程式 (例えば細くて一様な針金における熱伝導) に対する差分法について学んだ人を対象に、空間 2 次元の熱方程式 (ただし境界条件は同次 Dirichlet 境界条件とする) に対する差分法を説明し、MATLAB プログラムを提示する。

(2024/8/24 追記) 非同次問題バージョン (桂田 [1])、非同次 Neumann 境界条件バージョン (桂田 [2]) を書いた。

1 はじめに

この問題については、ずいぶん前から常識的となっている知見が多いけれど、私が卒研を始めた 20 年位前 (1993 年？) は、整理された参考書が見つけにくかった (1 次元のことしか書いてなかったり、差分方程式自体は書いてあっても行列・ベクトル表現が無かったり、説明が数学的に明快でなかったり (定理であっても証明がないとか)、参考になるプログラムがなかったり…)。そうなった理由は、1 つには利用できるコンピューター環境がまだまだ貧弱であったせいかもしれない。現在でも和書だとあまり事情は変わっていない気もするが (良さそうなのが出ても入手しにくくなったり…)、それは少し情けないですね。

そこで、本に書いてあることを一つ一つ解読して、プログラムを作成し、基本的な事実を数値実験で確認する、というのをしばらく (数年) 学部卒件のテーマとしたことがある (長方形領域の熱方程式については、桂田 [3] がまとめ)。今回紹介するのは、その際に分かったこと (整理したこと) のエッセンスと、それに基づく MATLAB プログラムである。一気に説明するが、実は相当圧縮してあるので、本当に納得しようとすると結構長い時間がかかると思う。

今日は、とりあえず以下のことをお話として理解してくれれば良い。

(i) 差分方程式については、1 次元の場合の差分方程式の自然な拡張である。

(ii) 差分方程式は連立 1 次方程式であるから、線形代数で学ぶような $Ax = b$ という形に書ける。もう少し詳しく書くと、第 n ステップから第 $n+1$ ステップに進めるのに、$A\boldsymbol{U}^{n+1} = B\boldsymbol{u}^n$ という形の方程式を解くことになる。A, B は n に依らないので、一度 $A = PLU$ と LU 分解しておくと、$A^{-1}b = U\backslash(L\backslash(Pb))$ で無駄なく計算できる。— 以上は 1 次元と同様と言えば同様であるが、実際に A, B を求めるのはそんなに簡単ではない。

(iii) 連立 1 次方程式の係数行列 A は帯行列である[1]。そのこと (成分に 0 が多い) を利用すると、効率的に解ける。
(これは今回は説明しないが参考まで: 未知数の個数はほぼ $N = N_x N_y$ であるが、Gauss の消去法で LU 分解を計算する場合、計算量は $O(N^3) = O(N_x^3 N_y^3)$ でなく、$O(N_x N_y^3)$ で済む。)

[1] 実は正値対称でもある。また B もそういう性質を持っている。

(iv) MATLAB を用いると、効率的なプログラムが簡潔で見通しよく書ける。
特に行列の Kronecker 積 $\otimes$ を用いると

$$\begin{aligned}(1)\qquad &\left[I_{N_x-1}\otimes I_{N_y-1}+\theta\lambda_y I_{N_x-1}\otimes K_{N_y-1}+\theta\lambda_x K_{N_x-1}\otimes I_{N_y-1}\right]\boldsymbol{U}^{n+1}\\ &=\left[I_{N_x-1}\otimes I_{N_y-1}-(1-\theta)\lambda_y I_{N_x-1}\otimes K_{N_y-1}-(1-\theta)\lambda_x K_{N_x-1}\otimes I_{N_y-1}\right]\boldsymbol{U}^{n}.\end{aligned}$$

(I_ℓ は単位行列で、K_ℓ は $\begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}$)

I_ℓ や K_ℓ を疎行列として与えると、MATLAB はそのことを利用した計算をしてくれる。

上の式 (1) を見ても、「複雑だ」と感じるかもしれないが、黒板に大きな行列を書き、C 言語で長いプログラムを書いていた頃を覚えている私にとっては、きれいな式に感じられる。

行列の Kronecker 積とは　$A=(a_{ij})\in\mathbf{R}^{m\times n}$, $B\in\mathbf{R}^{k\times\ell}$ に対して、

$$A\otimes B:=\begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}$$

で定まる $A\otimes B\in\mathbf{R}^{mk\times n\ell}$ を A と B の **Kronecker 積**あるいは**テンソル積**と呼ぶ (伊理 [4], [5] などを見よ)。

念のため注意　2 次元長方形領域における熱方程式を、この文書で説明する方法で解くのが得策かは良く分からない。最近では ADI 法で解いている人が多いような気がする。(ADI 法について解説すべきであるが、私はあまり詳しくない。) 一方で、この文書で説明した話の Poisson 方程式バージョンは、今でも使われているのかもしれない。

2　ふたたび

3　モデル問題

考える領域は長方形

$$\Omega:=(0,W)\times(0,H)=\left\{(x,y)\in\mathbf{R}^2\;\middle|\;0<x<W,\ 0<y<H\right\}$$

とする (長方形の内部)。$\partial\Omega$ はその境界 (長方形の 4 つの辺) を表すとする。

Ω における熱方程式の初期値 Dirihlet 境界値問題

$$\begin{aligned}&(2a)\qquad u_t(x,y,t)=\triangle u(x,y,t)\quad((x,y)\in\Omega,\ t>0),\\ &(2b)\qquad u(x,y,t)=0\quad((x,y)\in\partial\Omega,\ t>0),\\ &(2c)\qquad u(x,y,0)=f(x,y)\quad((x,y)\in\overline{\Omega})\end{aligned}$$

を考える。

実は、この問題は 2 次元の Fourier 級数を用いて解析的に解くことが出来る。

$$u(x,y,t)=\sum_{m=1}^{\infty}\sum_{n=1}^{\infty}b_{mn}e^{-\left((m/W)^2+(n/H)^2\right)\pi^2 t}\sin\frac{m\pi x}{W}\sin\frac{n\pi y}{H},$$

$$b_{mn}=\frac{4}{WH}\int_0^W\int_0^H f(x,y)\sin\frac{m\pi x}{W}\sin\frac{n\pi y}{H}dxdy.$$

これだけで分かることもあり、これだけでは分からないこともある (だから数値計算したい) のは 1 次元と同様である。

4 差分方程式

以下では、図を描いたりするのをさぼっているので、図を手で描き込んだりして下さい。

変数 (x,y,t) が動く範囲 $\overline{\Omega}\times[0,\infty)=[0,W]\times[0,H]\times[0,\infty)$ を差分格子に切る。
$N_x,N_y\in\mathbf{N}$, $\tau>0$ に対して、

$$h_x:=\frac{W}{N_x},\quad h_y:=\frac{H}{N_y},$$

$$x_i=ih_x\quad(i=0,1,\cdots,N_x),\quad y_j=jh_y\quad(j=0,1,\cdots,N_y),\quad t_n=n\tau\quad(n=0,1,\cdots)$$

とおき、(x_i,y_j,t_n) を格子点と呼ぶ。
$u^n_{i,j}:=u(x_i,y_j,t_n)$ $(0\le i\le N_x,\,0\le j\le N_y,\,n=0,1,\cdots)$ の近似値 $U^n_{i,j}$ を求めることを目標にする。$U^n_{i,j}$ は差分方程式の解として定義する。
熱方程式 $\dfrac{\partial u}{\partial t}=\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}$ において、t に関する微分係数を前進差分近似すると、陽解法の方程式

$$\frac{U^{n+1}_{i,j}-U^n_{i,j}}{\tau}=\frac{U^n_{i+1,j}-2U^n_{i,j}+U^n_{i-1,j}}{h_x^2}+\frac{U^n_{i,j+1}-2U^n_{i,j}+U^n_{i,j-1}}{h_y^2}\quad(1\le i\le N_x-1,\,1\le j\le N_y-1)$$

が得られる。ここでは初めから θ 法の差分方程式を考えることにする。

$$\begin{aligned}(3)\qquad \frac{U^{n+1}_{i,j}-U^n_{i,j}}{\tau}&=(1-\theta)\left(\frac{U^n_{i+1,j}-2U^n_{i,j}+U^n_{i-1,j}}{h_x^2}+\frac{U^n_{i,j+1}-2U^n_{i,j}+U^n_{i,j-1}}{h_y^2}\right)\\&\quad+\theta\left(\frac{U^{n+1}_{i+1,j}-2U^{n+1}_{i,j}+U^{n+1}_{i-1,j}}{h_x^2}+\frac{U^{n+1}_{i,j+1}-2U^{n+1}_{i,j}+U^{n+1}_{i,j-1}}{h_y^2}\right)\\&\quad(1\le i\le N_x-1,\,1\le j\le N_y-1).\end{aligned}$$

両辺に τ をかけて、

$$\lambda_x:=\frac{\tau}{h_x^2},\quad \lambda_y:=\frac{\tau}{h_y^2}$$

とおいて代入すると、

$$\begin{aligned}U^{n+1}_{i,j}-U^n_{i,j}&=(1-\theta)\left[\lambda_x\left(U^n_{i+1,j}-2U^n_{i,j}+U^n_{i-1,j}\right)+\lambda_y\left(U^n_{i,j+1}-2U^n_{i,j}+U^n_{i,j-1}\right)\right]\\&\quad+\theta\left[\lambda_x\left(U^{n+1}_{i+1,j}-2U^{n+1}_{i,j}+U^{n+1}_{i-1,j}\right)+\lambda_y\left(U^{n+1}_{i,j+1}-2U^{n+1}_{i,j}+U^{n+1}_{i,j-1}\right)\right].\end{aligned}$$

移項して ($U^{n+1}_{\circ\circ}$ は左辺、$U^n_{\circ\circ}$ は右辺)

$$\begin{aligned}&[1+2\theta(\lambda_x+\lambda_y)]U^{n+1}_{i,j}-\theta\lambda_x\left(U^{n+1}_{i+1,j}+U^{n+1}_{i-1,j}\right)-\theta\lambda_y\left(U^{n+1}_{i,j+1}+U^{n+1}_{i,j-1}\right)\\&\quad=[1-2(1-\theta)(\lambda_x+\lambda_y)]U^n_{i,j}+(1-\theta)\lambda_x\left(U^n_{i+1,j}+U^n_{i-1,j}\right)+(1-\theta)\lambda_y\left(U^n_{i,j+1}+U^n_{i,j-1}\right).\end{aligned}$$

コンパクトに表すため

$$b_x := -\theta\lambda_x, \quad b_y := -\theta\lambda_y, \quad a = 1 + 2\theta\,(\lambda_x + \lambda_y)\,,$$
$$c_x := (1-\theta)\lambda_x, \quad c_y := (1-\theta)\lambda_y, \quad d = 1 - 2(1-\theta)(\lambda_x + \lambda_y)$$

とおくと、

$$aU_{i,j}^{n+1} + b_x(U_{i+1,j}^{n+1} + U_{i-1,j}^{n+1}) + b_y(U_{i,j+1}^{n+1} + U_{i,j-1}^{n+1}) = dU_{i,j}^{n} + c_x(U_{i+1,j}^{n} + U_{i-1,j}^{n}) + c_y(U_{i,j+1}^{n} + U_{i,j-1}^{n}) \tag{4}$$
$$(1 \le i \le N_x - 1,\ 1 \le j \le N_y - 1).$$

境界条件からは

$$U_{i,j}^{n} = 0 \quad (i = 0, N_x \text{ で } 0 \le j \le N_y - 1,\ j = 0, N_y \text{ で } 0 \le i \le N_x - 1,\ n = 1, 2, \cdots). \tag{5}$$

初期条件からは

$$U_{i,j}^{0} = f(x_i, y_j) \quad (0 \le i \le N_x,\ 0 \le j \le N_y). \tag{6}$$

$U_{i,j}^{n}$ $(0 \le i \le N_x,\ 0 \le j \le N_y)$ が分かっているとき、(4) と (5) は $U_{i,j}^{n+1}$ $(0 \le i \le N_x,\ 0 \le j \le N_y)$ を未知数とする連立 1 次方程式である。各格子点に 1 つずつ 1 次方程式が乗っていると考えると分かりやすい (かも)。

5　差分方程式を行列とベクトルで表記する (とりあえず素朴に)

(簡単なはずだが、かなり手こずる。一つ一つ式で表して積み上げて行く。)

まず行列の記号の準備をする。$n \in \mathbf{N}$ とするとき、n 次単位行列を I_n と表す。$n \ge 2$ のとき、n 次正方行列 J_n, K_n を

$$J_n := \begin{pmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 1 & 0 \end{pmatrix}, \quad K_n := 2I_n - J_n = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

で定める。

(連立 1 次方程式を反復法で解く場合は必ずしもする必要がないが) 連立 1 次方程式 (4), (5) を解くために、$Ax = b$ (A は既知の行列, b は既知のベクトル, x が未知ベクトル) という形に書き表そう。

そのためには未知数を 1 次元的に並べてベクトルにする必要がある。

$n \ge 1$ とするとき、$U_{i,j}^{n}$ で $i \in \{0, N_x\}$ または $j \in \{0, N_y\}$ の場合は、Dirichlet 境界条件に由来して (5) $U_{i,j}^{n} = 0$ が成り立つので、$U_{i,j}^{n}$ $(1 \le i \le N_x - 1,\ 1 \le j \le N_y - 1)$ だけを未知数と考え、$1 \le i \le N_x - 1,\ 1 \le j \le N_y - 1$ に対して、

$$U_{\ell}^{n} := U_{i,j}^{n}, \quad \ell := j + (N_y - 1)(i - 1) \tag{7}$$

とおく (ℓ を 1 ずつ増加させると、(i, j) については 2 番めの添字 j が先に動くことに注意)。例えば $i, i \pm 1 \in \{1, \cdots, N_x - 1\}$, $j, j \pm 1 \in \{1, \cdots, N_y - 1\}$ ((x_i, y_j) の上下左右の格子点も領域内部にある) であれば、$s := N_y - 1$ とおくとき

$$aU_{\ell}^{n+1} + b_y(U_{\ell+1}^{n+1} + U_{\ell-1}^{n+1}) + b_x\left(U_{\ell+s}^{n+1} + U_{\ell-s}^{n+1}\right) = dU_{\ell}^{n} + c_y(U_{\ell+1}^{n} + U_{\ell-1}^{n}) + c_x(U_{\ell+s}^{n} + U_{\ell-s}^{n}).$$

$$N := (N_x - 1)(N_y - 1), \quad \boldsymbol{U}^n := \begin{pmatrix} U_1^n \\ U_2^n \\ \vdots \\ U_N^n \end{pmatrix}$$

とおくと、連立 1 次方程式は

$$A\boldsymbol{U}^{n+1} = B\boldsymbol{U}^n$$

となる。
ここで

$$A = \left(\begin{array}{ccccc|ccccc|ccccc|ccccc}
a & b_y & & & & b_x & & & & & & & & & & & & & & \\
b_y & a & b_y & & & & b_x & & & & & & & & & & & & & \\
 & \ddots & \ddots & \ddots & & & & \ddots & & & & & & & & & & & & \\
 & & b_y & a & b_y & & & & b_x & & & & & & & & & & & \\
 & & & b_y & a & & & & & b_x & & & & & & & & & & \\
\hline
b_x & & & & & & \ddots & & & & \ddots & & & & & & & & & \\
 & b_x & & & & & & \ddots & & & & \ddots & & & & & & & & \\
 & & \ddots & & & & & & \ddots & & & & \ddots & & & & & & & \\
 & & & b_x & & & & & & \ddots & & & & \ddots & & & & & & \\
 & & & & b_x & & & & & & & & & & & & & & & \\
\hline
 & & & & & & \ddots & & & & \ddots & & & & & b_x & & & & \\
 & & & & & & & \ddots & & & & \ddots & & & & & b_x & & & \\
 & & & & & & & & \ddots & & & & \ddots & & & & & \ddots & & \\
 & & & & & & & & & \ddots & & & & \ddots & & & & & b_x & \\
 & & & & & & & & & & & & & & & & & & & b_x \\
\hline
 & & & & & & & & & & b_x & & & & & a & b_y & & & \\
 & & & & & & & & & & & b_x & & & & b_y & a & b_y & & \\
 & & & & & & & & & & & & \ddots & & & & \ddots & \ddots & \ddots & \\
 & & & & & & & & & & & & & b_x & & & & b_y & a & b_y \\
 & & & & & & & & & & & & & & b_x & & & & b_y & a
\end{array}\right).$$

すなわち ($I := I_s$, $J := J_s$ として)

$$\begin{aligned}
A &= \begin{bmatrix}
aI + b_yJ & b_xI & & & \\
b_xI & aI + b_yJ & b_xI & & \\
 & \ddots & \ddots & \ddots & \\
 & & b_xI & aI + b_yJ & b_xI \\
 & & & b_xI & aI + b_yJ
\end{bmatrix} \\
&= \begin{pmatrix}
a & b_x & & \\
b_x & a & b_x & \\
 & \ddots & \ddots & \ddots \\
 & & b_x & a
\end{pmatrix} \otimes I + \begin{pmatrix}
1 & & & \\
 & 1 & & \\
 & & \ddots & \\
 & & & 1
\end{pmatrix} \otimes (b_yJ) \\
&= (aI_{N_x-1} + b_xJ_{N_x-1}) \otimes I_{N_y-1} + I_{N_x-1} \otimes (b_yJ_{N_y-1}) \\
&= ((1 + 2\theta(\lambda_x + \lambda_y))I_{N_x-1} - \theta\lambda_xJ_{N_x-1}) \otimes I_{N_y-1} + I_{N_x-1} \otimes (-\theta\lambda_yJ_{N_y-1}) \\
&= I_{N_x-1} \otimes I_{N_y-1} + \theta\lambda_yI_{N_x-1} \otimes (2I_{N_y-1} - J_{N_y-1}) + \theta\lambda_x(2I_{N_x-1} - J_{N_x-1}) \otimes I_{N_y-1} \\
&= I_{N_x-1} \otimes I_{N_y-1} + \theta\lambda_yI_{N_x-1} \otimes K_{N_y-1} + \theta\lambda_xK_{N_x-1} \otimes I_{N_y-1}.
\end{aligned}$$

同様にして

$$B = \begin{pmatrix} dI + c_yJ & c_xI & & & \\ c_xI & aI + c_yJ & c_xI & & \\ & \ddots & \ddots & \ddots & \\ & & c_xI & dI + c_yJ & c_xI \\ & & & c_xI & dI + c_yJ \end{pmatrix}$$

$$= \begin{pmatrix} d & c_x & & & \\ c_x & d & c_x & & \\ & \ddots & \ddots & \ddots & \\ & & c_x & d & \end{pmatrix} \otimes I + \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \otimes (c_yJ)$$

$$= (dI_{N_x-1} + c_xJ_{N_x-1}) \otimes I_{N_y-1} + I_{N_x-1} \otimes (c_yJ_{N_y-1})$$

$$= ((1 - 2(1-\theta)(\lambda_x + \lambda_y))I_{N_x-1} + (1-\theta)\lambda_xJ_{N_x-1}) \otimes I_{N_y-1} + I \otimes ((1-\theta)\lambda_yJ_{N_y-1})$$

$$= I_{N_x-1} \otimes I_{N_y-1} + \theta\lambda_yI_{N_x-1} \otimes (2I_{N_y-1} - J_{N_y-1}) + \theta\lambda_x(2I_{N_x-1} - J_{N_x-1}) \otimes I_{N_y-1}$$

$$= I_{N_x-1} \otimes I_{N_y-1} - (1-\theta)\lambda_yI_{N_x-1} \otimes K_{N_y-1} - (1-\theta)\lambda_xK_{N_x-1} \otimes I_{N_y-1}.$$

(もしも (7) でなく、

$$U_\ell^n := U_{i,j}^\ell, \quad \ell = i + (N_x - 1)(j-1)$$

によって $\{U_\ell^n\}$ を定義するならば、

$$A = I_{N_y-1} \otimes I_{N_x-1} + \theta\lambda_xI_{N_y-1} \otimes K_{N_x-1} + \theta\lambda_yK_{N_y-1} \otimes I_{N_x-1},$$
$$B = I_{N_y-1} \otimes I_{N_x-1} - (1-\theta)\lambda_xI_{N_y-1} \otimes K_{N_x-1} - (1-\theta)\lambda_yK_{N_y-1} \otimes I_{N_x-1}$$

となる。x と y を入れ替えるだけである。)

A で θ となっている部分を $-(1-\theta)$ に置き換えると B が得られる。

6 頭の整理

6.1 1 次元の復習

1 次元の復習から始める。$N \in \mathbf{N}$, $n := N - 1$, $h =$ 区間幅$/N$ とおく。同次 Dirichlet 境界条件 ($x = 0, 1$ で $u = 0$) を考えているとき、d^2/dx^2 を 2 階中心差分近似で離散化して現れる行列は

$$K' := \frac{1}{h^2} \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} = \frac{1}{h^2}(J_{N-1} - 2I_{N-1}) = -\frac{1}{h^2}K_{N-1}.$$

ゆえに θ 法の方程式は ($U_0^n = U_N^n = 0$ が成り立つとして)

$$\frac{1}{\tau}\left(\boldsymbol{U}^{n+1} - \boldsymbol{U}^n\right) = (1-\theta)K'\boldsymbol{U}^n + \theta K'\boldsymbol{U}^{n+1}.$$

移項して

$$(1 - \theta\tau K')\boldsymbol{U}^{n+1} = (1 + (1-\theta)\tau K')\boldsymbol{U}^n.$$

K_{N-1} で表せば

$$[1 + \theta\lambda K_{N-1}]\,\boldsymbol{U}^{n+1} = [1 - (1-\theta)\lambda K_{N-1}]\,\boldsymbol{U}^n.$$

左辺の行列で θ のところを $-(1-\theta)$ に換えたのが右辺の行列である (これはいつもそうなっていることである)。

6.2　2 次元

$N_x, N_y \in \mathbf{N}$, $h_x =$ 横幅$/N_x$, $h_y =$ 縦幅$/N_y$ とすると、$\triangle = \partial^2/\partial x^2 + \partial^2/\partial y^2$ を離散化した行列は

$$\begin{aligned} K' &:= \frac{J_{N_x-1} - 2I_{N_x-1}}{h_x^2} \otimes I_{N_y-1} + I_{N_x-1} \otimes \frac{J_{N_y-1} - 2I_{N_y-1}}{h_y^2} \\ &= -\frac{1}{h_x^2} K_{N_x-1} \otimes I_{N_y-1} - I_{N_x-1} \otimes \frac{1}{h_y^2} K_{N_y-1}. \end{aligned}$$

θ 法の方程式は 1 次元の場合と同じで

$$\frac{1}{\tau}\left(\boldsymbol{U}^{n+1} - \boldsymbol{U}^n\right) = (1-\theta)K'\boldsymbol{U}^n + \theta K'\boldsymbol{U}^{n+1}.$$

移項して

$$(1 - \theta\tau K')\boldsymbol{U}^{n+1} = (1 + (1-\theta)\tau K')\boldsymbol{U}^n.$$

$$\begin{aligned} &\left[I_{N_x-1} \otimes I_{N_y-1} + \theta\lambda_x K_{N_x-1} \otimes I_{N_y-1} + \theta\lambda_y I_{N_x-1} \otimes K_{N_y-1}\right] \boldsymbol{U}^{n+1} \\ &\quad = \left[I_{N_x-1} \otimes I_{N_y-1} - (1-\theta)\lambda_x K_{N_x-1} \otimes I_{N_y-1} - (1-\theta)\lambda_y I_{N_x-1} \otimes K_{N_y-1}\right] \boldsymbol{U}^n. \end{aligned}$$

7　MATLAB で 2 次元格子点を扱う

7.1　1 次元で肩ならし

区間 $[a, b]$ の n 等分点

$$x_i = a + (i-1)\frac{b-a}{n} \quad (i = 1, 2, \cdots, n+1)$$

で出来る $n+1$ 次元横ベクトルを求めるには `linspace(a,b,n+1)` とする。

MATLAB で区間を分割する仕方

```
>> linspace(0,1,5+1)
ans=
 0 0.2000 0.4000 0.6000 0.8000 1.0000
```

これを利用して 1 変数関数 $y = x^2 - 3x + 2$ $(-2 \leq x \leq 5)$ のグラフを描いてみよう。C 言語でプログラミングする場合の感覚だと

```
n=100;
x=linspace(-2,5,n+1);
y=zeros(n+1,1);
for i=1:n+1
  y(i)=x(i)^2-3*x(i)+2;
end
plot(x,y);
```

のようなループを用いたコードを書いてしまいそうだが、ベクトル演算を利用して

```
n=100;
x=linspace(-2,5,n+1);
y=x.^2-3*x+2;
plot(x,y);
```

とすると効率が良い。ベクトル x の各成分を 2 乗するのに x.^2 としているのが注意すべきところである。

かけ算、割り算以外では、普通に x の式を書いて構わない。例えば sin のグラフが描きたければ、単に plot(x,sin(x)) で良い。

7.2 では 2 変数関数のグラフを描いてみよう

2 変数関数 $f(x,y)$ の、格子点 (x_i, y_j) $(i = 1, \ldots, N_x{+}1;\ j = 1, \ldots, N_y{+}1)$ における値 $\{f(x_i, y_j)\}_{\substack{1\le i\le N_x+1\\ 1\le j\le N_y+1}}$ を求めるのに、やはり C 言語のプログラムの発想をすると、

```
fs=zeros(Nx+1,Ny+1);
for i=1:Nx+1
  for j=1:Ny+1
     fs(i,j)=f(x(i),y(j));
  end
end
```

のようなコードが考えられる。

しかし、1 次元の場合と同様にこのやり方は (特に二重ループになるので) 避けたい。

以下に説明する MATLAB で推奨されている方法は、初めて見るときは異様な感じがするかもしれないが (私はびっくりしました)、良く考えてみると納得出来る (今でもスマートとは思えないが、効率を上げるためにはそれなりに筋が通っている)。

まず各格子点 $\{(x_i, y_j)\}_{\substack{1\le i\le N_x+1\\ 1\le j\le N_y+1}}$ の x 座標, y 座標を記録した 2 次元配列を用意する。それ自体にループを使いたくなるが、MATLAB ではそれをするための関数 meshgrid() が用意されている。

格子点での x 座標, y 座標を用意する

```
>> nx=5; ny=10;
>> X=linspace(0,1,nx+1); Y=linspace(0,1,ny+1);
>> [x,y]=meshgrid(X,Y)

x =
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000

y =
         0         0         0         0         0         0
    0.1000    0.1000    0.1000    0.1000    0.1000    0.1000
    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000
    0.3000    0.3000    0.3000    0.3000    0.3000    0.3000
    0.4000    0.4000    0.4000    0.4000    0.4000    0.4000
    0.5000    0.5000    0.5000    0.5000    0.5000    0.5000
    0.6000    0.6000    0.6000    0.6000    0.6000    0.6000
    0.7000    0.7000    0.7000    0.7000    0.7000    0.7000
    0.8000    0.8000    0.8000    0.8000    0.8000    0.8000
    0.9000    0.9000    0.9000    0.9000    0.9000    0.9000
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

x も y も $(10+1, 5+1)$ 型の (2 次元) 配列である。一般に (N_y+1, N_x+1) 型の配列が出来るわけである (順番に注意せよ)。

$f(x,y) = (x-1/2)^2 + (y-1/2)^2$ のグラフを描くには

```
>> mesh(x,y,(x-0.5).*(x-0.5)+(y-0.).*(y-0.5))
>> mesh(x,y,(x-0.5).^2+(y-0.5).^2)
```

とする。x や y は行列と解釈出来るので、かけ算やべき乗を成分毎に行うために . をつけた演算子 .* や .^ を用いるのは、1 次元の場合の例と同様である。

7.3 2 次元配列をベクトルにする, その逆

連立 1 次方程式を扱うとき、(x_i, y_j) における u の値を集めた数列 (2 次元配列) $\{u(x_i, y_j)\}$ を 1 次元的に並べたベクトルを作る必要が生じる。

MATLAB では配列の次元を変更するための関数 reshape() が用意されている。(m,n) 型の配列 $u = \{u_{ij}\}$ を mn 次元の縦ベクトルにするには、reshape(u,m*n,1) とすれば良い (実は u(:) という手もあるが、一つ覚えるならば、reshape() の方が応用が効きそうなので、それがオススメ)。

注意すべきは、MATLAB では、メモリーの中では、u_{11}, u_{21}, $\cdots$, u_{m1}, u_{12}, u_{22}, $\cdots$, u_{m2}, u_{13}, u_{23}, $\cdots$, u_{m3}, $\cdots$, u_{1n}, u_{2n}, $\cdots$, u_{mn} の順に格納されていること、いわゆる “column major order” で添字が動くことである (Fortran ユーザーを念頭に開発されたからであろう。“row major order” に慣れている C 言語ユーザーには、違和感があるかもしれないが、仕方がない。)。

以下は、次節のプログラム heat2d.m に現れる reshape() の解説である。

2 次元配列を 1 次元配列にする

```
U=reshape(u[2:Ny,2:Nx],(Nx-1)*(Ny-1),1);
```

u は 2 次元配列 $[\mathtt{u}(j,i)]_{\substack{1\le j\le Ny+1\\ 1\le i\le Nx+1}}$ で、u(j,i) は $U^n_{i-1,j-1}$ である。つまり添字の番号がずれてはいるが、u は $\left\{U^n_{ij}\right\}_{\substack{0\le i\le Nx\\ 0\le j\le Ny}}$ を並べたものである。u(2:Ny,2:Nx) は、$\left\{U^n_{ij}\right\}_{\substack{1\le i\le Nx-1\\ 1\le j\le Ny-1}}$ を並べた 2 次元配列である。それを reshape(·,(Nx-1)*(Ny-1),1) することで、$\boldsymbol{U}^n$ が得られる。

逆変換は次のようにする。

1 次元配列を 2 次元配列に戻す

```
u(2:Ny,2:Nx)=reshape(U,Ny-1,Nx-1);
```

U は $\{U^{n+1}_{ij}\}_{\substack{1\le i\le Nx-1\\ 1\le j\le Ny-1}}$ を 1 次元的に並べたものになっている。それを reshape(·,Ny-1,Nx-1) とすることで、2 次元配列になっている。それを u の対応する場所 u(2:Ny,2:Nx) に上書きコピーしている。

余談 7.1 (私はたくさん混乱しました) 実は、ここに書いたことは混乱を生じさせやすい。一度理解した後も、うろ覚えで作業すると、よく間違えた。MATLAB は Fortran と同じで配列が column major order である、というのは間違いなく覚えているのだが、meshgrid() の生成する 2 次元配列がひねってあるせいだろう。meshgrid() を使わずに、自分でコーディングするとそういう順番にはしないので、meshgrid() を使ったときと、そうでないときに食い違いが生じる。自分でたたき台にするプログラムを用意して、それを一度きちんと理解して、後は真似をして作業するものだろうか？ ■

8 MATLAB プログラム

係数行列を用意するプログラムは独立させた。

heat2d.m[2], heat2d_mat.m[3] を入手して (~/Documents/MATLAB などに置き)、コマンド・ウィンドウで

```
>> heat2d
```

とすれば実行できる。

[2] http://nalab.mind.meiji.ac.jp/labo/text/heat2d.m
[3] http://nalab.mind.meiji.ac.jp/labo/text/heat2d_mat.m

heat2d.m

```
% heat2d.m
% 長方形領域における熱方程式 u_t=△u（同次 Dirichlet 境界条件）を差分法で解く
% 入手 curl -O https://m-katsurada.sakura.ne.jp/program/fdm/heat2d.m
% 解説 https://m-katsurada.sakura.ne.jp/labo/text/heat2d.pdf
function heat2d
  a=0; b=2; c=0; d=1;
  Nx=50;
  Ny=50;
  hx=(b-a)/Nx;
  hy=(d-c)/Ny;
  theta=0.5;
  tau=0.5/(1/hx^2+1/hy^2); % 陽解法の安定性条件（こんなに小さくなくても OK）
  lambdax=tau/hx^2;
  lambday=tau/hy^2;

% 差分方程式 A U^{n+1}=B U^n の行列
  [A,B]=heat2d_mat(Nx,Ny,lambdax,lambday,theta);
% 格子点の座標ベクトル x=(x_1,x_2,...,x_{Nx+1}), y=(y_1,y_2,...,y_{Ny+1})
  X=linspace(a,b,Nx+1);
  Y=linspace(c,d,Ny+1);
% 格子点の x,y 座標の配列 X={X_{ij}}, Y={Y_{ij}}
  [x,y]=meshgrid(X,Y);
% 初期値 sin(pi x) sin(pi y)
  u=sin(pi*x) .* sin(pi*y);
    %
  msg = sprintf("Nx=%d, Ny=%d, τ=%e, θ=%f, λx=%f, λy=%f",...
  Nx, Ny, tau, theta, lambdax, lambday);
  disp(msg);
% 初期値のグラフを描く
  mesh(x,y,u);
  disp('初期値を書きました。何かキーを押してください。')
  pause
% 径数行列の LU 分解
  [AL,AU,ap]=lu(A,'vector');

  Tmax=1;
  disp('繰り返し')
  dt=0.005;
  skip=dt/tau;
  U=reshape(u(2:Ny,2:Nx),(Nx-1)*(Ny-1),1);
  Nmax = ceil(Tmax / tau);
  for n=1:Nmax
    t=n*tau;
    U=B*U;
    U=AU\(AL\U(ap,:));
    if mod(n,skip)==0
      u(2:Ny,2:Nx)=reshape(U,Ny-1,Nx-1);
      msg=sprintf("t=%f", t);
      disp(msg);
      meshc(x,y,u);
      axis([a b c d -1 1]);
      drawnow;
    end
  end
```

heat2d_mat.m

```
% heat2d_mat.m
% 長方形領域における熱方程式 u_t=△u (Dirichlet 境界条件) を解くための差分方程式
% 入手 curl -O https://m-katsurada.sakura.ne.jp/program/fdm/heat2d_mat.m
% 解説 https://m-katsurada.sakura.ne.jp/labo/text/heat2d.pdf
%   A U^{n+1}=B U^n
% の行列 A, B を求める。 (2015/5/1 作成, 2015/5/31 コメント修正)
% Nx=3; Ny=3; hx=1/Nx; hy=1/Ny; theta=0.5; tau=0.5/(1/hx^2+1/hy^2); lamx=tau/hx^2; lamy=tau/hy^2;
% heat2d_mat(Nx,Ny,lamx,lamy,theta)
% A =
%    1.5000   -0.1250   -0.1250         0
%   -0.1250    1.5000         0   -0.1250
%   -0.1250         0    1.5000   -0.1250
%         0   -0.1250   -0.1250    1.5000
% B =
%    0.5000    0.1250    0.1250         0
%    0.1250    0.5000         0    0.1250
%    0.1250         0    0.5000    0.1250
%         0    0.1250    0.1250    0.5000
function [A,B]=heat2d_mat(Nx,Ny,lambdax,lambday,theta)
  Ix=speye(Nx-1,Nx-1);
  Iy=speye(Ny-1,Ny-1);
  vx=ones(Nx-2,1);
  Jx=sparse(diag(vx,1)+diag(vx,-1));
  vy=ones(Ny-2,1);
  Jy=sparse(diag(vy,1)+diag(vy,-1));
  Kx=2*Ix-Jx;
  Ky=2*Iy-Jy;
% x major (y changes first, l=j+(Ny-1)*i)
  A=kron(Ix,Iy)+theta*lambday*kron(Ix,Ky)+theta*lambdax*kron(Kx,Iy);
  B=kron(Ix,Iy)-(1-theta)*lambday*kron(Ix,Ky)-(1-theta)*lambdax*kron(Kx,Iy);
% y major (x chandes first, l=i+(Nx-1)*j)
%  A=kron(Iy,Ix)+theta*lambdax*kron(Iy,Kx)+theta*lambday*kron(Ky,Ix);
%  B=kron(Iy,Ix)-(1-theta)*lambdax*kron(Iy,Kx)-(1-theta)*lambday*kron(Ky,Ix);
```

9 非同次境界条件の場合

境界条件 (2b) を非同次な

$$u(x,y,t)=b(x,y) \quad ((x,y)\in\partial\Omega,\ t>0) \tag{8}$$

や

$$u(x,y,t)=b(x,y,t) \quad ((x,y)\in\partial\Omega,\ t>0) \tag{9}$$

で置き換えた問題については、別文書 [1] で扱うことにする。

参考文献

[1] 桂田祐史：2 次元熱方程式の非同次 Dirichlet 境界値問題を解く差分法プログラムを作る, https://m-katsurada.sakura.ne.jp/labo/text/heat2d-nonhomo.pdf (2024 年〜).

[2] 桂田祐史：非同次 Neumann 境界条件下の熱方程式に対する差分法 — MATLAB を使って数値計算 —, https://m-katsurada.sakura.ne.jp/labo/text/heat2n-nonhomo.pdf (2024/8/18, 2024/8/24).

[3] 桂田祐史：熱方程式に対する差分法 I — 区間における熱方程式 —, https://m-katsurada.sakura.ne.jp/labo/text/heat-fdm-1.pdf (1998 年〜).

[4] 伊理正夫：一般線形代数, 岩波書店 (2003), 伊理正夫, 線形代数 I, II, 岩波講座応用数学, 岩波書店 (1993, 1994) の単行本化.

[5] 伊理正夫：線形代数汎論, 朝倉書店 (2009), 「一般線形代数」のリニューアル.