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

桂田 祐史

2024年8月16日書き始め, 2024年8月17日公開, 2024年9月10日

目 次

1 はじめに

ずっと以前に、卒研で差分法によるシミュレーションをしていて、C で差分法のプログラムを色々作ったのだけれど ([1])、最近はすっかりごぶさたしている。MATLAB, Python, Julia など便利なものが普及してきたのを横目で見て、「そういうのを使えば便利だよな」と考えてはみたけれど…同次境界条件の場合のプログラムを MATLAB で書いたところで、ストップ ([2], [3] — 2015年頃の話だ)。ゼミのネタにならないものに時間を割く訳には行かない。

一応、非同次方程式の差分法プログラムも、[1] には C 言語で示してあるけれど、自分でも MATLAB とかで書かれた使えるプログラムがあれば、そちらの採用から考えたくなる。

という訳で、今回、久しぶりにそういうのをやってみる気になった。目標を一言でまとめると: 非同次境界条件の差分方程式を解くプログラムをまずは MATLAB で作成する。Dirichlet 境界条件、それから Neumann 境界条件、それから Poisson 方程式に移り、それから Python あるいは Julia. (多分最後までは行かなくて、途中で打ち切りになるだろうけれど。Poisson 方程式まではなんとかやりたいな。)

しばらく余談　なぜこのようなプログラムを作る気になったか、動機を述べておきたい。

私はある講義 (「応用複素関数」) で、ポテンシャル問題 (Laplace 方程式の境界値問題) の解説をしていて、学生には FreeFem++ というソフトウェアを勧めている。

FreeFem++ は有限要素法にもとづき、色々な偏微分方程式の問題を手軽に解けるので大変ありがたい。レポート課題として、

$$\triangle\phi = 0 \quad \text{in } \Omega, \qquad \frac{\partial\phi}{\partial n} = V_n \quad \text{in } \partial\Omega$$

を解くことで、非圧縮ポテンシャル流のシミレーションをしなさい、という問題を出した。

V_n は速度 $\boldsymbol{v}$ の外向き単位法線方向の成分である ($V = \boldsymbol{v}\cdot\boldsymbol{n}$)。このことからこの問題は物理的に鮮明なイメージがついて、楽しく遊べる。数学的にも解が存在するためには $\displaystyle\int_{\partial\Omega} V_n\, d\sigma = 0$ という整合条件を満たさなくてはいけないので、自分で選んでチェックしないといけないとか、色々やるべきことがある。

これは手頃な問題と考えていたのだが、Ω として矩形領域 (辺が座標軸に平行な長方形領域) を選んで、Python による差分法プログラムで解決しようとした学生が数人いた。最初は、ああ、そう来たかと思った。差分法では、矩形領域以外で解くことが難しいということもあったが、まあ、矩形領域以外を選べと書いておかなかったからな (失敗、失敗)。まあ、Python のプログラム例が集まるのも良いことかも。

ところがレポートを読み始めて頭を抱えた。次のような残念なプログラムが多かった。

- Dirichlet 境界値問題を解いている (全員！それでは指定した問題が解けない…当然結果が全然違っている)。
- 連立 1 次方程式を Gauss-Seidel 法のような定常反復法で解いている (今どき定常問題 (時刻変数を含まない問題) をそんな方法で解くの？消去法や CG 法に勝てるはずがなくて、それを選ぶ理由がないと思うのだけど…一体何を参考にしたのだろう)。

ネットで探してみたけれど、意外と差分法の Python プログラムは探しにくいと感じた。自分でそこまでやるかは分からないけれど、MATLAB プログラムを作っておけば Python にするのは簡単だろう (それこそ AI が翻訳してくれるかもしれないし)。

2　1 次元の問題の復習

1 次元の熱方程式の非同次 Dirichlet 境界値問題の差分方程式については

「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

の第 1 章に説明がある。

$$A = (1+2\theta\lambda) - \theta\lambda J, \quad B = [1-2(1-\theta)\lambda]\, I + (1-\theta)\lambda J, \quad \boldsymbol{U}^n = (U_1^n, \cdots, U_{N-1}^n)^\top$$

として次の差分方程式が得られる。

$$A\boldsymbol{U}^{n+1} = \begin{pmatrix} [1-2(1-\theta)\lambda]\, U_1^n + (1-\theta)\lambda(U_0^n + U_2^n) \\ [1-2(1-\theta)\lambda]\, U_2^n + (1-\theta)\lambda(U_1^n + U_3^n) \\ \vdots \\ [1-2(1-\theta)\lambda]\, U_{N-2}^n + (1-\theta)\lambda(U_{N-3}^n + U_{N-1}^n) \\ [1-2(1-\theta)\lambda]\, U_{N-1}^n + (1-\theta)\lambda(U_{N-2}^n + U_N^n) \end{pmatrix} + \begin{pmatrix} \alpha\theta\lambda \\ 0 \\ \vdots \\ 0 \\ \beta\theta\lambda \end{pmatrix} \tag{1}$$

(λ, θ, J, I が何かなどの説明はサボる。)

α, β は境界値 ($U(0,t) = \alpha$, $U(1,t) = \beta$) ということであった。つまり、この方程式は実は

$$A\boldsymbol{U}^{n+1} = B\boldsymbol{U}^n + \begin{pmatrix} (1-\theta)\lambda U_0^n \\ 0 \\ \vdots \\ 0 \\ (1-\theta)\lambda U_N^n \end{pmatrix} + \begin{pmatrix} \theta\lambda U_0^{n+1} \\ 0 \\ \vdots \\ 0 \\ \theta\lambda U_N^{n+1} \end{pmatrix}$$

ということである ($U_0^n = U_0^{n+1} = \alpha$, $U_N^n = U_N^{n+1} = \beta$ を代入すると (1) が得られる)。

境界条件 $u(0,t) = \alpha$, $u(1,t) = \beta$ を仮定しているので、結局は

$$A\boldsymbol{U}^{n+1} = B\boldsymbol{U}^n + \begin{pmatrix} \lambda\alpha \\ 0 \\ \vdots \\ 0 \\ \lambda\beta \end{pmatrix}$$

となるわけだが、境界条件が時間依存して $u(0,t) = \alpha(t)$, $u(1,t) = \beta(t)$ となっている場合は

$$A\boldsymbol{U}^{n+1} = B\boldsymbol{U}^n + \begin{pmatrix} (1-\theta)\lambda\alpha(t_n) + \theta\lambda\alpha(t_{n+1}) \\ 0 \\ \vdots \\ 0 \\ (1-\theta)\lambda\beta(t_n) + \theta\lambda\beta(t_{n+1}) \end{pmatrix}$$

を使うことになる。

3　2 次元の問題の復習

3.1　[1] から一歩進める

記述の簡単化のため、まず、2 次元矩形領域 Ω で、Dirichlet 境界値が時間依存しない場合、つまり $u(x,y,t) = b(x,y)$ という境界条件の場合を考える。

$$u_t(x,y,t) = \triangle u(x,y,t) \quad ((x,y) \in \Omega,\ t > 0) \tag{2a}$$

$$u(x,y,t) = b(x,y) \quad ((x,y) \in \partial\Omega) \tag{2b}$$

$$u(x,y,0) = f(x,y) \quad ((x,y) \in \overline{\Omega}) \tag{2c}$$

繰り返しになるが、一応 [1] にはどのように差分方程式を作るか説明をした上で、C 言語のプログラムを示してある。しかし差分方程式を式として提示していないので、MATLAB プログラムにすぐ書き換える訳にはいかない。

以下、記号は [1] に準じる。$W > 0$, $H > 0$, $\Omega = (0, W) \times (0, L)$, $N_x \in \mathbb{N}$, $N_y \in \mathbb{N}$, $h_x = W/N_x$, $h_y = H/h_y$, $x_i = ih_x$ $(i = 0, 1, \cdots, N_x)$, $y_j = jh_y$ $(j = 0, 1, \cdots, N_y)$, $\tau > 0$, $t_n = n\tau$ $(n = 0, 1, 2, \cdots)$, $u_{ij}^n = u(x_i, y_j, t_n)$ $(0 \le i \le N_x,\ 0 \le j \le N_y,\ n = 0, 1, \cdots)$.

u_{ij}^n の近似 U_{ij}^n を求めるための、いわゆる θ 法の差分方程式は導出してある。θ は $0 \le \theta \le 1$ を満たすパラメーターである。

各々の n に対して、U_{ij}^n $(1 \le i \le N_x - 1,\ 1 \le j \le N_y - 1)$ から 1 つのベクトル $\boldsymbol{U}^n = (U_1^n, \cdots, U_N^n)$ を作る。

$$U_\ell^n = U_{i,j}^n, \quad \ell = i + (N_x - 1)(j - 1) \quad (1 \le i \le N_x - 1,\ 1 \le j \le N_y - 1) \tag{3}$$

$N = (N_x - 1)(N_y - 1)$ である。

- $j = 1$; $i = 1, \cdots, N_x - 1$ に対して $\lambda_y b(x_i, y_{j-1}) = \lambda_y b(x_i, y_0)$ が右辺に足される。
- $j = N_y - 1$; $i = 1, \cdots, N_x - 1$ に対して $\lambda_y b(x_i, y_{j+1}) = \lambda_y b(x_i, y_{N_y})$ が右辺に足される。
- $i = 1$; $j = 1, \cdots, N_y - 1$ に対して $\lambda_x b(x_{i-1}, y_j) = \lambda_x b(x_0, y_j)$ が右辺に足される。
- $i = N_x - 1$; $j = 1, \cdots, N_y - 1$ に対して $\lambda_x b(x_{i+1}, y_j) = \lambda_x b(x_N, y_j)$ が右辺に足される。

A, B を [2] で説明した行列として、次のようになる。

$$A\boldsymbol{U}^{n+1} = B\boldsymbol{U}^n + \lambda_y \begin{pmatrix} b(x_1, y_0) \\ b(x_2, y_0) \\ \vdots \\ b(x_{N_x-2}, y_0) \\ b(x_{N_x-1}, y_0) \\ \hline 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \hline \vdots \\ \hline \vdots \\ \hline b(x_1, y_{N_y}) \\ b(x_2, y_{N_y}) \\ \vdots \\ b(x_{N_x-2}, y_{N_y}) \\ b(x_{N_x-1}, y_{N_y}) \end{pmatrix} + \lambda_x \begin{pmatrix} b(x_0, y_1) \\ 0 \\ \vdots \\ 0 \\ b(x_{N_x}, y_1) \\ \hline b(x_0, y_2) \\ 0 \\ \vdots \\ 0 \\ b(x_{N_x}, y_2) \\ \hline \vdots \\ \hline \vdots \\ \hline b(x_0, y_{N_y-1}) \\ 0 \\ \vdots \\ 0 \\ b(x_{N_x}, y_{N_y-1}) \end{pmatrix} \begin{matrix} \leftarrow i = 1, j = 1 \\ \leftarrow i = 2, j = 1 \\ \vdots \\ \leftarrow i = N_x - 2, j = 1 \\ \leftarrow i = N_x - 1, j = 1 \\ \leftarrow i = 1, j = 2 \\ \leftarrow i = 2, j = 2 \\ \vdots \\ \leftarrow i = N_x - 2, j = 2 \\ \leftarrow i = N_x - 1, j = 2 \\ \vdots \\ \vdots \\ \leftarrow i = 1, j = N_y - 1 \\ \leftarrow i = 2, j = N_y - 1 \\ \vdots \\ \leftarrow i = N_x - 2, j = N_y - 1 \\ \leftarrow i = N_x - 1, j = N_y - 1 \end{matrix} \tag{4}$$

[1] では、この連立 1 次方程式をコンピューター上に構成するプログラムを提示したが、式 (4) そのものは提示しなかった。

境界値が時間依存する場合は、

- 右辺第 2 項のベクトル内の $b(x_i, y_j)$ を $(1-\theta)\lambda_y b(x_i, y_j, t_n) + \theta\lambda_y b(x_i, y_j, t_{n+1})$ に置き換える
- 右辺第 3 項のベクトル内の $b(x_i, y_j)$ を $(1-\theta)\lambda_x b(x_i, y_j, t_n) + \theta\lambda_x b(x_i, y_j, t_{n+1})$ に置き換える

3.2 MATLAB の mesh() のため (4) を Row major に直す

前項では、二重添え字の持つ数列 U_{ij}^n からベクトル $\boldsymbol{U}^n = (U_\ell^n)$ を作るのに、$\ell = i + (j-1)N_x$ という対応を用いた。これは 2 次元配列をいわゆる “column major” で扱うのに近い。[1] でそうした理由は当時参照したテキストの多くでそうなっていたことに起因すると思う (もう記憶が定かでない)。

さて「2 次元配列を MATLAB は column major で扱う」というのは常識だけれど、meshgrid で可視化したりする場合は、row major のように扱う方が都合が良いように思われる (なぜ MATLAB がそうなっているのか、全く理解できない)。

そこでここでは前項の説明の row major バージョンを述べる。

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

(6)
$$A\boldsymbol{U}^{n+1} = B\boldsymbol{U}^n + \lambda_x \begin{pmatrix} b(x_0, y_1) \\ b(x_0, y_2) \\ \vdots \\ b(x_0, y_{N_y-2}) \\ b(x_0, y_{N_y-1}) \\ \hline 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \hline \vdots \\ \hline \vdots \\ \hline b(x_{N_x}, y_1) \\ b(x_{N_x}, y_2) \\ \vdots \\ b(x_{N_x}, y_{N_y-2}) \\ b(x_{N_x}, y_{N_y-1}) \end{pmatrix} + \lambda_y \begin{pmatrix} b(x_1, y_0) \\ 0 \\ \vdots \\ 0 \\ b(x_1, y_{N_y}) \\ \hline b(x_2, y_0) \\ 0 \\ \vdots \\ 0 \\ b(x_2, y_{N_y}) \\ \hline \vdots \\ \hline \vdots \\ \hline b(x_{N_x-1}, y_0) \\ 0 \\ \vdots \\ 0 \\ b(x_{N_x-1}, y_{N_y}) \end{pmatrix} \begin{matrix} \leftarrow i = 1, j = 1 \\ \leftarrow i = 1, j = 2 \\ \vdots \\ \leftarrow i = 1, j = N_y - 2 \\ \leftarrow i = 1, j = N_y - 1 \\ \leftarrow i = 2, j = 1 \\ \leftarrow i = 2, j = 2 \\ \vdots \\ \leftarrow i = 2, j = N_y - 2 \\ \leftarrow i = 2, j = N_y - 1 \\ \vdots \\ \vdots \\ \leftarrow i = N_x - 1, j = 1 \\ \leftarrow i = N_x - 1, j = 2 \\ \vdots \\ \leftarrow i = N_x - 1, j = N_y - 2 \\ \leftarrow i = N_x - 1, j = N_y - 1 \end{matrix}$$

境界値が時間依存する場合は、

- 右辺第 2 項のベクトル内の $b(x_i, y_j)$ を $(1-\theta)\lambda_x b(x_i, y_j, t_n) + \theta\lambda_x b(x_i, y_j, t_{n+1})$ に置き換える
- 右辺第 3 項のベクトル内の $b(x_i, y_j)$ を $(1-\theta)\lambda_y b(x_i, y_j, t_n) + \theta\lambda_y b(x_i, y_j, t_{n+1})$ に置き換える

4 同次 Dirichlet 境界条件の問題の MATLAB プログラム

桂田 [2] で説明した、矩形領域 Ω における熱方意識の初期値境界値問題（ただし境界条件は同次 Dirichlet 境界条件)

(7a) $$u_t(x, y, t) = \triangle u(x, y, t) \quad ((x, y) \in \Omega,\ t > 0),$$

(7b) $$u(x, y, t) = 0 \quad ((x, y) \in \partial\Omega, t > 0),$$

(7c) $$u(x, y, 0) = f(x, y) \quad ((x, y) \in \overline{\Omega})$$

を解く MATLAB プログラムを再録する。
(2 つのプログラムに分けてあるが、現在の MATLAB は 1 つのファイルに複数の関数を収められるので、1 つにまとめてしまうのもありかもしれない。)

4.1 heat2d_mat.m

以下に示すのは、連立 1 次方程式の径数行列を計算するプログラムである。これは非同次境界条件の場合にも無修正で利用できる。

```
% 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);
```

4.2 heat2d.m

こちらがシミュレーション・プログラムで、中から heat2d_mat() を呼び出している。

$$\Omega = (0,2) \times (0,1), \quad f(x,y) = \sin(\pi x)\sin(\pi y).$$

```
% 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, λ x=%f, λ y=%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
```

4.3 入手・コンパイル・実行

まずターミナルで

```
curl -O https://m-katsurada.sakura.ne.jp/program/fdm/heat2d.m
```

これを実行するには、例えば

```
cp -p heat2d.m heat2d_mat.m ~/Documents/MATLAB
```

とコピーして (私はシンボリック・リンクをはることにしている)、MATLAB の中で

```
>> heat2d
```

(分割数はデフォールトの $N_x = 50$, $N_y = 50$ が採用される。)

あるいは、分割数 N_x, N_y を指定して

```
>> heat2d_nh(100,100)
```

とする。

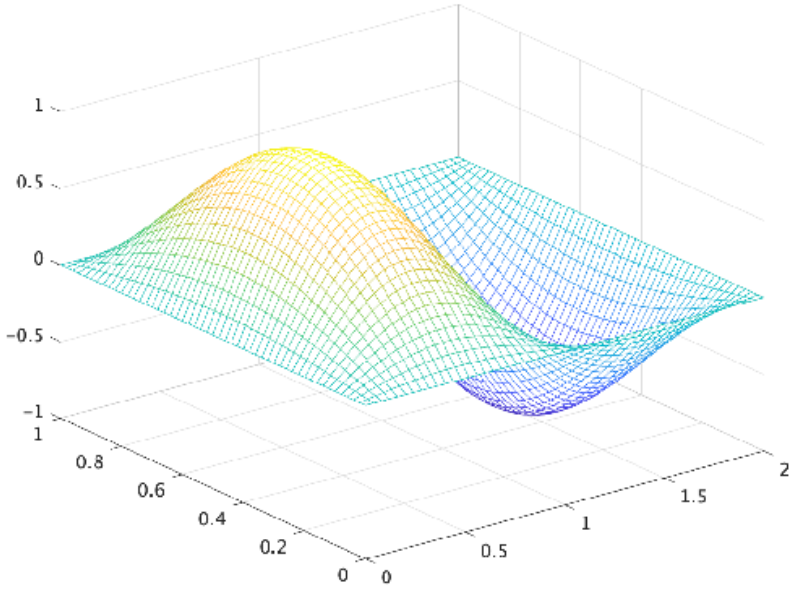

図 1: `heat2d.m` 初期値のグラフ

4.3.1 実行結果

$0 \leq t \leq \texttt{Tmax} = 0.5$ における誤差 (max error) は次のようになった。$O(h_x^2 + h_y^2)$ となっていると考えられる。

N_x	N_y	max error
50	50	4.099456e-04
100	100	1.028208e-04
200	200	2.571340e-05

表 1: 分割を細かくしていったとき ($N_x = N_y$ を 50, 100, 200) の誤差の変化

5 非同次 Dirichlet 境界条件の MATLAB プログラム (定常な境界値の場合)

5.1 heat2d_nh.m

出来立て。実は結構デバッグに苦労したので、まだあまり自信がない。一応、厳密解の分かる問題を 1 つ解いて誤差を測ってみて、小さいことを確認してある。

$$u_t(x,y,t) = \triangle u(x,y,t) \quad ((x,y) \in \Omega,\ t > 0), \tag{8a}$$

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

$$u(x,t,0) = f(x,y) \quad ((x,y) \in \overline{\Omega}) \tag{8c}$$

ただし

$$\Omega = (0,1) \times (0,1), \tag{9}$$

$$b(x,y) = (x-1/2)^2 - (y-1/2)^2, \tag{10}$$

$$f(x,y) = b(x,y) + \sin(\pi x)\sin(2\pi y). \tag{11}$$

この問題は自分にとって都合が良いように作った。境界値である b は、その定義式のままで Ω で調和関数であることがトリックである (これは熱方程式と境界条件を満たす特解である)。初期値はそれに $\sin(\pi x)\sin(2\pi y)$ という摂動を与えたもので、$\sin(\pi x)\sin(2\pi y)$ は同次 Dirichlet 境界条件の元での Laplacian の固有関数なので、Fourier の方法による解の公式が即座に適用できる。実際、次の関数が厳密解である。

$$u(x,y,t) = b(x,y) + e^{-5\pi^2 t}\sin(\pi x)\sin(2\pi y). \tag{12}$$

```
% heat2d_nh.m
% 長方形領域における熱方程式 u_t=△u (非同次 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_nh.m
% 解説 https://m-katsurada.sakura.ne.jp/labo/text/heat-nonhomo.pdf
% 一応の完成をして、heat2d_nh.m として公開した (2024/8/17)。少し推敲 (2024/8/24)。

function heat2d_nh(Nx, Ny)
  arguments
    Nx=50;
    Ny=50;
  end
  % Dirichlet 境界条件の場合の係数行列
  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;
    % row 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);
    % column 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);
  end
```

```
% 長方形領域 (a,b) × (c,d)
a=0; b=1; c=0; d=1;
% Dirichlet 境界値 b() 調和関数を選ぶとこれが平衡解 (定常解)
function val = dbv(x,y)
  val = (x-0.5).*(x-0.5)-(y-0.5).*(y-0.5);
end
% 初期値 (平衡解を境界値が 0 のもので摂動する)
function val = iv(x,y)
  val = dbv(x,y) + sin(pi*x).*sin(2*pi*y);
end
% 厳密解が分かる
function val = exact(x,y,t)
  val = dbv(x,y) + exp(- 5 * pi * pi * t) * sin(pi*x).*sin(2*pi*y);
end
% 長方形の下の辺での b()
function val = bbottom(x)
  val = dbv(x,c);
end
% 長方形の上の辺での b()
function val = btop(x)
  val = dbv(x,d);
end
% 長方形の左の辺での b()
function val = bleft(y)
  val = dbv(a, y);
end
% 長方形の右の辺での b()
function val = bright(y)
  val = dbv(b, y);
end
% 誤差を測るためのノルム (最大値ノルム)
function val = supnorm(a)
  [m,n]=size(a);
  val = norm(reshape(a,m*n,1),Inf);
end
% 格子への分割
% 分割数 Nx, Ny は関数引数とした。プログラム先頭の arguments ブロックを見よ。
%Nx=100;
%Ny=100;
hx=(b-a)/Nx;
hy=(d-c)/Ny;
theta=0.5;
tau=0.5/(1/hx^2+1/hy^2); % 陽解法の場合のギリギリの刻み (陰解法なので大きくても OK)
lambdax=tau/hx^2;
lambday=tau/hy^2;
disp(sprintf('Nx=%d, Ny=%d, hx=%f, hy=%f, tau=%e, λ x=%f, λ y=%f', ...
      Nx, Ny, hx, hy, tau, lambdax, lambday));

% 差分方程式 A U^{n+1}=B U^n の行列
[A,B]=heat2d_mat(Nx,Ny,lambdax,lambday,theta);
% LU 分解
[AL,AU,ap]=lu(A,'vector');

% 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
u= iv(x,y);
% 極限 (境界値 b の式が調和ならば)
ulimit = dbv(x,y);
```

```
  % 初期値のグラフを描く
  disp('初期値のグラフ')
  meshc(x,y,u); axis([a b c d -2 2]); drawnow; shg;

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

  % 境界値 左の辺、右の辺
  blvector=bleft(Y(2:Ny));
  brvector=bright(Y(2:Ny));
  lindex=1:Ny-1;
  rindex=(Nx-2)*(Ny-1)+1:(Nx-1)*(Ny-1);
  % 境界値 下の辺、上の辺
  bbvector=bbottom(X(2:Nx));
  btvector=btop(X(2:Nx));
  bindex=1:Ny-1:(Nx-1)*(Ny-1);
  tindex=Ny-1:Ny-1:(Nx-1)*(Ny-1);

  % どこまで計算するか
  Tmax=0.5;
  Nmax = ceil(Tmax/tau);
  % グラフを表示する時間の間隔
  dt=0.005;
  skip=dt/tau;
  %
  disp('ループに入ります。何かキーを押してください。')
  pause
  t=0;
  maxerror=0;
  for n=0:Nmax
    % 連立1次方程式の右辺の計算
    U=B*U;
    % 移項
    U(lindex)=U(lindex)+lambdax*blvector;
    U(rindex)=U(rindex)+lambdax*brvector;
    U(bindex)=U(bindex)+lambday*bbvector;
    U(tindex)=U(tindex)+lambday*btvector;
    % 連立1次方程式を解いて差分解を求める
    U=AU\(AL\U(ap,:));
    t=(n+1)*tau;
    if mod(n,skip)==0
      u(2:Ny,2:Nx)=reshape(U,Ny-1,Nx-1);
      meshc(x,y,u); axis([a b c d -2 2]); drawnow; shg;
      error = supnorm(u-exact(x,y,t));
      if error > maxerror maxerror = error; end
      disp(sprintf('t=%f, ||error||=%e, ||u||=%e, ||u ∞||=%e',...
     t, error, supnorm(u), supnorm(ulimit)));
    end
  end
  display(sprintf('||u(・,t)-u(・, ∞)||=%e', supnorm(u-ulimit)));
  display(sprintf('Nx=%d,Ny=%d,max error=%e', Nx, Ny, maxerror));
end
```

5.2 入手・コンパイル・実行

まずターミナルで

```
curl -O https://m-katsurada.sakura.ne.jp/program/fdm/heat2d_nh.m
```

これを実行するには、安直にやるなら

```
cp -p heat2d_nh.m ~/Documents/MATLAB
```

とコピーして (私はシンボリック・リンクをはることにしている)、MATLAB の中で

```
>> heat2d_nh
```

$0 \le t \le$ Tmax $= 0.5$ における誤差 (max error) は次のようになった。$O(h_x^2 + h_y^2)$ となっていると考えられる。

N_x	N_y	max error
50	50	4.099456e-04
100	100	1.028208e-04
200	200	2.571340e-05

表 2: 分割を細かくしていったとき ($N_x = N_y$ を 50, 100, 200) の誤差の変化

A 混乱回避のため Row major, Column major の解説

最近は、かえって混乱しそうな解説が増えてきたような気がする。Wikipedia によると (Wikipedia を頼るようになっている…)

> In computing, row-major order and column-major order are methods for storing multidimensional arrays in linear storage such as random access memory.

A.1 Column major とは

例えば Fortran 言語で書かれたプログラムで、`real a(3,2)` と変数宣言すると、メモリー中で `a(1,1)`, `a(2,1)`, `a(3,1)`, `a(1,2)`, `a(2,2)`, `a(3,2)` の順に並ぶ。このように列の中で連続している要素が隣接していることを Column major (「列優先」?) という。

(必ずしもメモリー内の配置の話をしている訳ではないのだが) 私が桂田 [1] で選んだ 2 種類の 1 次元化

- $\ell = j(N_x + 1) + i$ $(0 \le i \le N_x,\ 0 \le j \le N_y)$
- $\ell = (j-1)(N_x + 1) + i - 1$ $(1 \le i \le N_x - 1,\ 1 \le j \le N_y - 1)$

というのは Column major 風と言えるであろう。C でプログラムを書いていたのに、当時読んだテキストに影響されたのか…(笑)

A.2 Row major とは

例えば C 言語で書かれたプログラムで、`double a[3][2];` と変数宣言すると、メモリー中で `a[0][0]`, `a[0][1]`, `a[0][2]`, `a[0][0]`, `a[0][1]`, `a[0][2]` の順に並ぶ。このように行の中で連続している要素が隣接していることを Row major (「行優先」?) という。

A.3 プログラミング言語・ライブラリィごとの違い

- Column major であるもの: Fortran 以外に、MATLAB とその親戚 (GNU Octave, Scilab), Julia, R, S-Plus, Eigen などなど。
- Row major であるもの: C の親戚 (C++, Objective C), Pascal, PL/I, NumPy などなど。

注: Python には、普通のプログラミング言語でいう配列は存在しない (それらしく見えるのはリストであり、linear storage ではない)。だから、ここには登場しない。Python でよく使われる Numpy の ndarray というのだったっけ？それは配列で、Row major order である。

```
>>> a=np.array([1,2,3,4,5,6]);
>>> a.shape
(6,)
>>> a.shape=2,3;
>>> a
array([[1, 2, 3],
       [4, 5, 6]])
>>> a.shapge=3,2
>>> a
array([[1, 2],
       [3, 4],
       [5, 6]])
>>> a=np.array([1,2,3,4,5,6]);
>>> a.reshape(2,3)
array([[1, 2, 3],
       [4, 5, 6]])
>>> a.reshape(2,3,order='F')
array([[1, 3, 5],
       [2, 4, 6]])
>>> a.reshape(6,)
array([1, 2, 3, 4, 5, 6])
```

注: Eigen は、デフォールトは column major order だが (https://eigen.tuxfamily.org/dox/group__TopicStorageOrders.html)、Row major, Column major, 好きな方を選べる。

```
typedef Matrix<double, 3, 3, RowMajor> matrm;
typedef Matrix<double, 3, 3, ColMajor> matcm;
matrm a;
matcm b;
```

A.4 MATLAB の mesh() (不思議な話)

```
L=3; H=2;
a=-L; b=L; c=-H; d=H;
nx=30; ny=20;
X=linspace(a,b,nx+1);
Y=linspace(c,d,ny+1);
[x,y]=meshgrid(X,Y);
% f(x,y)=sin(x)sin(2y)
F=sin(x).*sin(2*y);
figure('Name','graph of F')
meshc(x,y,F)
fig1=gcf; figure(fig1)
% g(x,y)=x^2-y^2
G=x.^2-y.^2;
figure('Name','graph of G')
meshc(x,y,G)
fig2=gcf; figure(fig2)
```

```
whos x y G
```

とすると、x, y, G はどれも 21x31 という Size が表示される。つまり (Ny+1)×(Nx+1) ということである。

そのため、MATLAB で mesh(), meshx() を使って可視化するには、$G(x_i, y_j)$ の値を G(j,i) に記録しておくと都合が良い？？？なんでこんなことになっているのか？？？？まあ分かっていればどうということはないが。

```
X=linspace(a,b,Nx+1);
Y=linspace(c,d,Ny+1);
[x,y]=meshgrid(X,Y);

u=cos(pi*x) .* cos(pi*y);
% ここまで普通と思われるが、こうしてから例えば
% U=reshape(u,(Nx+1)*(Ny+1),1);
% と 1 次元化すると、row major になってしまうわけだ。
% これは Neumann 境界値問題の場合で、Dirichlet 境界値問題の場合は
% 境界上の点を除いて 1 次元化するのが自然だが、その場合は次のように
% するのが自然だろう。
U=reshape(u(2:Ny,2:Nx),(Nx-1)*(Ny-1),1);
```

https://m-katsurada.sakura.ne.jp/labo/text/new-intro-matlab/node47.html

参考文献

[1] 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〜).

[2] 桂田祐史：長方形領域における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/heat2d.pdf (2015 年〜).

[3] 桂田祐史：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.pdf (2015 年〜).