2次元Poisson非同次Dirichlet境界値問題を解く差分法プログラム(MATLAB)

桂田 祐史

2024年8月24日書き始め, 2024年8月25日公開, 2024年8月29日

目 次

1 はじめに

(準備中)

ものすごく手短に言うと、はるか昔に桂田 [2] というのを書いたけれど、今便利なコードを持っておこう、と考えた。

2 差分方程式の導出

つい最近、熱方程式に対する差分法の解説を書いた (桂田 [1])。そのマイナー・チェンジ・バージョンである。

2.1 桂田 [1] の §3.1 の真似をして作る

2次元矩形領域 Ω で、Dirichlet 境界値が $D(x,y)$ という境界条件の場合を考える。

$$-\triangle u(x,y) = f(x,y) \quad ((x,y) \in \Omega) \tag{1a}$$

$$u(x,y,t) = D(x,y) \quad ((x,y) \in \partial\Omega). \tag{1b}$$

$W > 0$, $H > 0$, $\Omega = (0, W) \times (0, L)$, $f \colon \Omega \to \mathbb{R}$. $D \colon \partial\Omega \to \mathbb{R}$.

$N_x \in \mathbb{N}$, $N_y \in \mathbb{N}$, $h_x = W/N_x$, $h_y = H/N_y$, $x_i = ih_x$ $(i = 0, 1, \cdots, N_x)$, $y_j = jh_y$ $(j = 0, 1, \cdots, N_y)$, $u_{ij} = u(x_i, y_j)$ $(0 \le i \le N_x, 0 \le j \le N_y\mathrm{n})$. $f_{i,j} = f(x_i, y_j)$.

u_{ij} の近似 U_{ij} を求めるための差分方程式として

$$-\left(\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h_x^2} + \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h_y^2}\right) = f_{i,j}.$$

を採用する。

$$\alpha = \frac{2}{h_x^2} + \frac{2}{h_y^2}, \quad \beta_x = \frac{1}{h_x^2}, \quad \beta_y = \frac{1}{h_y^2}$$

とおくと

$$\alpha U_{i,j} - \beta_x(U_{i+1,j} + U_{i-1,j}) - \beta_y(U_{i,j+1} + U_{i,j-1}) = f_{i,j}.$$

領域内部にある格子点に対応する $U_{i,j}$ を未知数と考えるのが自然である。その個数は

$$N := (N_x - 1)(N_y - 1).$$

U_{ij} $(1 \le i \le N_x - 1, 1 \le j \le N_y - 1)$ から 1 つのベクトル $\boldsymbol{U}$ を作る。

(2a) $\quad U_\ell = U_{i,j} \quad \ell = i + (N_x - 1)(j - 1) \quad (1 \le i \le N_x - 1, 1 \le j \le N_y - 1),$

(2b) $\quad \boldsymbol{U} := (U_1, U_2, \cdots, U_N)^\top.$

同様に f_{ij} $(1 \le i \le N_x - 1, 1 \le j \le N_y - 1)$ から 1 つのベクトル $\boldsymbol{f}$ を作る。

(3a) $\quad f_\ell = f_{i,j} \quad \ell = i + (N_x - 1)(j - 1) \quad (1 \le i \le N_x - 1, 1 \le j \le N_y - 1),$

(3b) $\quad \boldsymbol{f} := (f_1, f_2, \cdots, f_N)^\top.$

境界上にある格子点での値 $U_{i,j}$ は、境界条件によって “既知” であるから、それを含む項は、方程式の右辺に移項する。

- $j = 1$; $i = 1, \cdots, N_x - 1$ に対して $\beta_y U_{i,j-1} = \beta_y D(x_i, y_0)$ が右辺に移項される。
- $j = N_y - 1$; $i = 1, \cdots, N_x - 1$ に対して $\beta_y U_{i,j+1} = \beta_y D(x_i, y_{N_y})$ が右辺に移項される。
- $i = 1$; $j = 1, \cdots, N_y - 1$ に対して $\beta_x U_{i-1,j} = \beta_x D(x_0, y_j)$ が右辺に移項される。
- $i = N_x - 1$; $j = 1, \cdots, N_y - 1$ に対して $\beta_x U_{i+1,y_j} = \beta_x D(x_N, y_j)$ が右辺に移項される。

連立 1 次方程式を行列・ベクトル表現しよう。係数行列を A とするとき、A がどういう形になるかよく知られているが、Kronecker 積を用いると、次のように表せる。

$$K_{N_x-1} := \frac{1}{h_x^2}(2I_{N_x-1} - J_{N_x-1}), \quad K_{N_y-1} := \frac{1}{h_y^2}(2I_{N_y-1} - J_{N_y-1}),$$

(4) $$A = I_{N_y-1} \otimes K_{N_x-1} + K_{N_y-1} \otimes I_{N_x-1}.$$

その辺の説明はどこかに書いたっけ？？

(1 次元 Laplacian の差分近似は (区間を N 等分するとき) K_{N-1} となるので、2 次元で $-\frac{\partial^2}{\partial x^2}$ の差分近似は $I_{N_y-1} \otimes K_{N_x-1}$ となるのが理解できると、$-\triangle = -\frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial y^2}$ の差分近似が A になることは分かりやすい。誰かがきちんと書くと良いのだけどね。)

熱方程式の場合は、行列がどうなるか地道に計算して示した上で、それが Kronecker 積で表したものと一致することは確かめてある (桂田 [3])。

上に書いた移項手順の結果、次の方程式が得られる。

$$
A\boldsymbol{U} = \boldsymbol{f} + \beta_y \begin{pmatrix} D(x_1,y_0) \\ D(x_2,y_0) \\ \vdots \\ D(x_{N_x-2},y_0) \\ D(x_{N_x-1},y_0) \\ \hline 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \hline \vdots \\ \hline \vdots \\ \hline D(x_1,y_{N_y}) \\ D(x_2,y_{N_y}) \\ \vdots \\ D(x_{N_x-2},y_{N_y}) \\ D(x_{N_x-1},y_{N_y}) \end{pmatrix} + \beta_x \begin{pmatrix} D(x_0,y_1) \\ 0 \\ \vdots \\ 0 \\ D(x_{N_x},y_1) \\ \hline D(x_0,y_2) \\ 0 \\ \vdots \\ 0 \\ D(x_{N_x},y_2) \\ \hline \vdots \\ \hline \vdots \\ \hline D(x_0,y_{N_y-1}) \\ 0 \\ \vdots \\ 0 \\ D(x_{N_x},y_{N_y-1}) \end{pmatrix} \begin{matrix} \leftarrow i=1, j=1 \\ \leftarrow i=2, j=1 \\ \vdots \\ \leftarrow i=N_x-2, j=1 \\ \leftarrow i=N_x-1, j=1 \\ \leftarrow i=1, j=2 \\ \leftarrow i=2, j=2 \\ \vdots \\ \leftarrow i=N_x-2, j=2 \\ \leftarrow i=N_x-1, j=2 \\ \vdots \\ \vdots \\ \leftarrow i=1, j=N_y-1 \\ \leftarrow i=2, j=N_y-1 \\ \vdots \\ \leftarrow i=N_x-2, j=N_y-1 \\ \leftarrow i=N_x-1, j=N_y-1 \end{matrix} \tag{5}
$$

2.2 MATLAB の mesh() のため (5) を Row major に直す

(これも [1] の §3.2 の真似である。)

前項では、二重添字を持つ数列 $U_{i,j}$ からベクトル $\boldsymbol{U} = (U_\ell)$ を作るのに、$\ell = i + (j-1)(N_x-1)$ 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。

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。

(ゴタゴタ書かないで、最初からこちらだけを書いて提示するべきかもしれないが、多分 MATLAB のこの変な事情がなければ、自分は前項の式を使っていると思うので、あえて両方書いておく次第である。式を書いておかないとすぐに使えないので。)

U_{ij} $(1 \le i \le N_x-1,\ 1 \le j \le N_y-1)$ から 1 つのベクトル $\boldsymbol{U}$ を作る (並べ順に注意)。

$$
U_\ell = U_{i,j} \quad \ell = j + (N_y-1)(i-1) \quad (1 \le i \le N_x-1,\ 1 \le j \le N_y-1), \tag{6a}
$$

$$
\boldsymbol{U} := (U_1, U_2, \cdots, U_N)^\top. \tag{6b}
$$

同様に f_{ij} $(1 \le i \le N_x-1,\ 1 \le j \le N_y-1)$ から 1 つのベクトル $\boldsymbol{f}$ を作る。

$$
f_\ell = f_{i,j} \quad \ell = j + (N_y-1)(i-1) \quad (1 \le i \le N_x-1,\ 1 \le j \le N_y-1), \tag{7a}
$$

$$
\boldsymbol{f} := (f_1, f_2, \cdots, f_N)^\top. \tag{7b}
$$

$$K_{N_x-1} := \frac{1}{h_x^2}(2I_{N_x-1} - J_{N_x-1}), \quad K_{N_y-1} := \frac{1}{h_y^2}(2I_{N_y-1} - J_{N_y-1})$$

はこれまでと同じ。次の Kronecker 積の順番に注意せよ。

$$A = K_{N_x-1} \otimes I_{N_y-1} + I_{N_x-1} \otimes K_{N_y-1}. \tag{8}$$

方程式の右辺は次式に示すようになる。当たり前であるが、どういう成分が現れるかは同じで、その並べ方だけが違う訳である。

$$A\boldsymbol{U} = \boldsymbol{f} + \beta_x \begin{pmatrix} D(x_0,y_1) \\ D(x_0,y_2) \\ \vdots \\ D(x_0,y_{N_y-2}) \\ D(x_0,y_{N_y-1}) \\ \hline 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \hline \vdots \\ \hline \vdots \\ \hline D(x_{N_x},y_1) \\ D(x_{N_x},y_2) \\ \vdots \\ D(x_{N_x},y_{N_y-2}) \\ D(x_{N_x},y_{N_y-1}) \end{pmatrix} + \beta_y \begin{pmatrix} D(x_1,y_0) \\ 0 \\ \vdots \\ 0 \\ D(x_1,y_{N_y}) \\ \hline D(x_2,y_0) \\ 0 \\ \vdots \\ 0 \\ D(x_2,y_{N_y}) \\ \hline \vdots \\ \hline \vdots \\ \hline D(x_{N_x-1},y_0) \\ 0 \\ \vdots \\ 0 \\ D(x_{N_x-1},y_{N_y}) \end{pmatrix} \begin{matrix} \leftarrow i=1, j=1 \\ \leftarrow i=1, j=2 \\ \vdots \\ \leftarrow i=1, j=N_y-2 \\ \leftarrow i=1, j=N_y-1 \\ \leftarrow i=2, j=1 \\ \leftarrow i=2, j=2 \\ \vdots \\ \leftarrow i=2, j=N_y-2 \\ \leftarrow i=2, j=N_y-1 \\ \vdots \\ \vdots \\ \leftarrow i=N_x-1, j=1 \\ \leftarrow i=N_x-1, j=2 \\ \vdots \\ \leftarrow i=N_x-1, j=N_y-2 \\ \leftarrow i=N_x-1, j=N_y-1 \end{matrix} \tag{9}$$

2.2.1 行列 A を MATLAB で作るコード `poisson2d_mat.m`

次が自作コード。

poisson2d_mat.m

```
% poisson2d_mat.m
% 長方形領域における Poisson 方程式 -△ u=f (Dirichlet B.C.) を解く差分法の行列
% 入手 curl -O https://m-katsurada.sakura.ne.jp/program/fdm/poisson2d_mat.m
% 解説 https://m-katsurada.sakura.ne.jp/labo/text/poisson2d-nonhomo.pdf
%   A U=f
% の行列 A を求める (2024/8/24 作成)。
function A = poisson2d_mat(W,H,Nx,Ny)
  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));
  hx = W / Nx;
  hy = H / Ny;
  betax = 1.0 / hx^2;
  betay = 1.0 / hy^2;
  Kx=betax * (2*Ix - Jx);
  Ky=betay * (2*Iy - Jy);
  % row major (y changes first, l=j+(Ny-1)*(i-1))
  A=kron(Kx,Iy) + kron(Ix,Ky);
  % column major (x chandes first, l=i+(Nx-1)*(j-1))
  %  A=kron(Iy,Kx) + kron(Ky,Ix);
end
```

次はずっと以前 (2017 年？) に、何かを見て作ったもの (丸ごとパクリかもしれないが、古いことなので出典を覚えていない)。

poisson_coef.m

```
function A=poisson_coef(W, H, nx, ny)
% 長方形領域 (0,W) × (0,H) における Poisson 方程式の Dirichlet 境界値問題
% Laplacian を差分近似した行列を求める。
% 長方形を nx × ny 個の格子に分割して差分近似する。
% MATLAB では
%  (1) 行列は Fotran と同様の column first であり、
%  (2) mesh(), contour() による「行列描画」は Z(j,i) と添字の順が普通と逆なので、
% l=i+(j-1)*(nx-1) と row first となるように 1 次元的番号付けする
  hx=W/nx;
  hy=H/ny;
  m=nx-1;
  n=ny-1;
  ex=ones(nx,1);
  ey=ones(ny,1);
  Lx=spdiags([-ex,2*ex,-ex],-1:1,m,m)/(hx*hx);
  Ly=spdiags([-ey,2*ey,-ey],-1:1,n,n)/(hy*hy);
  A=kron(speye(m,m),Ly)+kron(Lx,speye(n,n));
```

個人的に、spdiags() の文法が覚えきれないので (まあ、検索すれば良いのだけど、大事なことは覚えているものでやるべきだと思っている)、kron() 使う方が好みである。

3 MATLAB プログラムによる数値実験

3.1 poisson2d_nh.m

厳密解の分かる問題を 1 つ解いて誤差を測ってみて、小さいことを確認する。

$$-\triangle u(x,y) = f(x,y) \quad ((x,y) \in \Omega), \tag{10a}$$

$$u(x,y) = D(x,y) \quad ((x,y) \in \partial\Omega). \tag{10b}$$

ただし

$$\Omega = (0,1) \times (0,1), \tag{11}$$

$$u(x,y) = \cos(\pi x)\cos(\pi y) \quad ((x,y) \in \overline{\Omega}), \tag{12}$$

$$D(x,y) = u(x,y) = \cos(\pi x)\cos(\pi y) \quad ((x,y) \in \partial\Omega), \tag{13}$$

$$f(x,y) = -\triangle u(x,y) = 2\pi^2 \cos(\pi x)\cos(\pi y) \quad ((x,y) \in \Omega). \tag{14}$$

(最初に厳密解 u を決めて、それから D と f を求めた。)

次のプログラムでは、境界値 D を与える関数を `b_l()`, `b_r()`, `b_b()`, `b_t()` と 4 つに分けてある。1 つですませる方が自然かもしれない。Neumann 境界条件の場合は角点の問題が生じるので、分けざるを得なくて、それに合わせたため、こうなっている。

```
% poisson2d_nh.m
% 長方形領域における Poisson -△ u=f (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/poisson2d_nh.m
% 解説 https://m-katsurada.sakura.ne.jp/labo/text/poisson2d-nonhomo.pdf
% 一応の完成をして、poisson2d_nh.m として公開した (2024/8/25)。

function poisson2d_nh(Nx,Ny)
  arguments
    Nx=50
    Ny=50
  end
  % Dirichlet 境界条件の場合の係数行列
  function A = poisson2d_mat(W,H,Nx,Ny)
    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));
    hx = W / Nx;
    hy = H / Ny;
    betax = 1.0 / hx^2;
    betay = 1.0 / hy^2;
    Kx=betax * (2*Ix - Jx);
    Ky=betay * (2*Iy - Jy);
    % row major (y changes first, l=j+(Ny-1)*(i-1))
    A=kron(Kx,Iy) + kron(Ix,Ky);
    % column major (x chandes first, l=i+(Nx-1)*(j-1))
    %  A=kron(Iy,Kx) + kron(Ky,Ix);
  end
  function A=poisson_coef(W, H, nx, ny)
    hx=W/nx;
    hy=H/ny;
    m=nx-1;
    n=ny-1;
    ex=ones(nx,1);
    ey=ones(ny,1);
    Lx=spdiags([-ex,2*ex,-ex],-1:1,m,m)/(hx*hx);
    Ly=spdiags([-ey,2*ey,-ey],-1:1,n,n)/(hy*hy);
    A=kron(speye(m,m),Ly)+kron(Lx,speye(n,n));
  end
```

```
% 長方形領域 (a,b) × (c,d)
a=0; b=1; c=0; d=1;
%
function u = exact_solution(x, y)
  u = cos(pi * x) .* cos(pi * y);
end
% Poisson 方程式の右辺 (exact_solution の -△)
function val = f(x, y)
  val = 2 * pi^2 * cos(pi * x) .* cos(pi * y);
end
% 長方形の下の辺での b()
function val = b_b(x)
  val = exact_solution(x,c);
end
% 長方形の上の辺での b()
function val = b_t(x)
  val = exact_solution(x,d);
end
% 長方形の左の辺での b()
function val = b_l(y)
  val = exact_solution(a, y);
end
% 長方形の右の辺での b()
function val = b_r(y)
  val = exact_solution(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
%Nx=100;
%Ny=100;
hx=(b-a)/Nx;
hy=(d-c)/Ny;
betax=1.0/hx^2;
betay=1.0/hy^2;
disp(sprintf('Nx=%d, Ny=%d, hx=%f, hy=%f, betax=%f, betay=%f', ...
      Nx, Ny, hx, hy, betax, betay));

% 差分方程式 A U^{n+1}=B U^n の行列
A=poisson2d_mat(b-a,d-c,Nx,Ny);
N=(Nx-1)*(Ny-1);
if N <= 20
  disp('A='); full(A)
end
%A2=poisson_coef(b-a,d-c,Nx,Ny);
%err=A-A2;
%disp(sprintf('A-A2=%e', supnorm(err)));
%clear A2 err
% 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);

% Poisson 方程式の右辺の関数値 (連立 1 次方程式に必要)
u=f(x,y);
meshc(x,y,u); axis([a b c d -50 50]); title('graph of f'); drawnow; shg
```

```
  % 計算に用いるため、f の内部格子点での値をベクトル U に格納する
  U=reshape(u(2:Ny,2:Nx),(Nx-1)*(Ny-1),1);

  % 境界値（左の辺、右の辺）
  blvector=b_l(Y(2:Ny));
  brvector=b_r(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=b_b(X(2:Nx));
  btvector=b_t(X(2:Nx));
  bindex=1:Ny-1:(Nx-1)*(Ny-1);
  tindex=Ny-1:Ny-1:(Nx-1)*(Ny-1);

  % 境界条件により分かっている値を右辺に移項
  U(lindex)=U(lindex)+betax*blvector;
  U(rindex)=U(rindex)+betax*brvector;
  U(bindex)=U(bindex)+betay*bbvector;
  U(tindex)=U(tindex)+betay*btvector;

  % 連立 1 次方程式を解いて差分解を求める
  U=AU\(AL\U(ap,:));
  % 差分解を meshgrid に収める
  u(2:Ny,2:Nx)=reshape(U,Ny-1,Nx-1);
  % 境界条件を用いて境界上の格子点での値を求める
  u(:,1)=b_l(Y);
  u(:,Nx+1)=b_r(Y);
  u(1,:)=b_b(X);
  u(Ny+1,:)=b_t(X);
  % 解のグラフ
  figure; meshc(x,y,u); axis([a b c d -2 2]);
  title('graph of the approximate solution');
  drawnow;
  shg;
  disp('何かキーを押して下さい。');
  % 誤差を調べる
  error = supnorm(u-exact_solution(x,y));
  disp(sprintf('Nx=%d, Ny=%d, ||error||=%e, ||u||=%e',...
        Nx, Ny, error, supnorm(u)));
  % 比較のため厳密解を表示する
  meshc(x,y,exact_solution(x,y)); axis([a b c d -2 2]);
  title('graph of the exact solution'); drawnow; shg
end
```

3.2 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/poisson2d_nh.m
```

これを実行するには、例えば

```
cp -p poisson2d_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

```
>> poisson2d_nh
```

(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

```
>> poisson2d_nh(100,100)
```

とする。

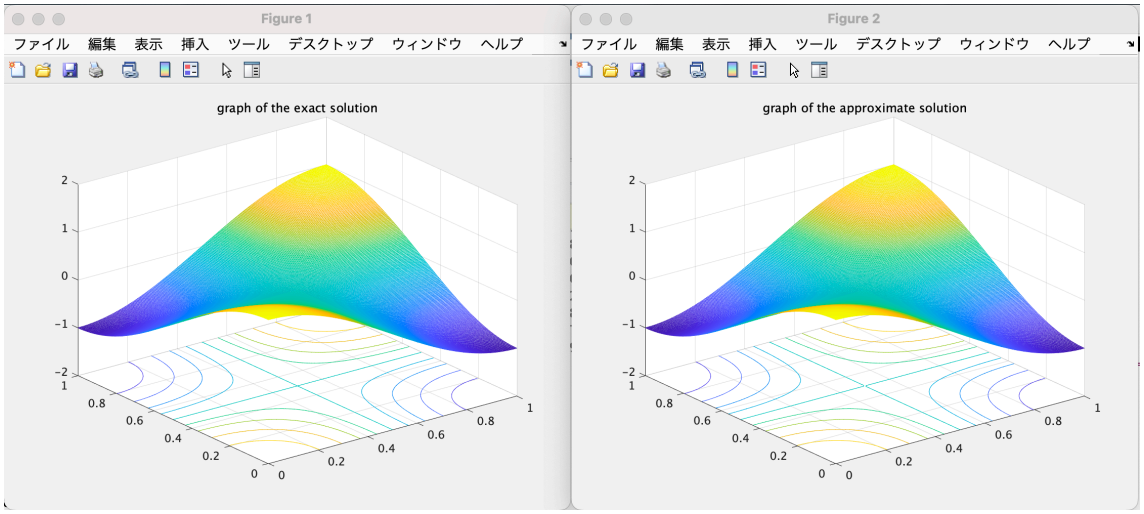

図 1: $Nx = Ny = 200$ 厳密解と差分解 (違いが見て分かるはずはないが…)

誤差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ている。

```
>> poisson2d_nh
Nx=50, Ny=50, hx=0.020000, hy=0.020000, betax=2500.000000, betay=2500.000000
Nx=50, Ny=50, ||error||=5.924640e-05, ||u||=1.000000e+00

>> poisson2d_nh(100,100)
Nx=100, Ny=100, hx=0.010000, hy=0.010000, betax=10000.000000, betay=10000.000000
Nx=100, Ny=100, ||error||=1.482114e-05, ||u||=1.000000e+00

>> poisson2d_nh(200,200)
Nx=200, Ny=200, hx=0.005000, hy=0.005000, betax=40000.000000, betay=40000.000000
Nx=200, Ny=200, ||error||=3.705884e-06, ||u||=1.000000e+00

>> poisson2d_nh(400,400)
Nx=400, Ny=400, hx=0.002500, hy=0.002500, betax=160000.000000, betay=160000.000000
Nx=400, Ny=400, ||error||=9.265347e-07, ||u||=1.000000e+00
```

A MATLAB メモ

- figure を全て閉じる。`close all hidden`

- drawnow は、“Figure を更新してコールバックを処理する”。要するに確実に画面に表示させる。
- shg は現在の figure を表示する。裏に回さず、確実に画面で見えるようにしてくれる。ひどい短縮の仕方だ。
- デフォールトの引数

```
function poisson2d_nh(Nx, Ny)
  arguments
    Nx=50
    Ny=50
  end
  ...
```

(まあ、分かるんだけど、行数が増えるな。)

- 確認のため、値を表示させるとき、“ans =” は分かりにくいので、それっぽい名前の変数に代入して、clear する、という手がある。

```
u_value=u
clear u_value
```

大きいとメモリーと時間の無駄遣いだから、あまりしたくない。

```
disp('u=')
disp(u)
```

とするのかな。次のようなことは u が配列だとできない。

```
disp(['u=' u])
```

不自由だね。

- 何だ？

```
文字ベクトルの終端が正しくありません。
```

文字列が閉じていないよ (' を忘れていない？)、という意味だ。

- 1 つのファイル内に複数の関数を書けるようになった。それは便利なようだが、うっかりローカルな変数のつもりで、グローバルな変数に代入して値を壊すというううっかりミスを避ける方法はあるのか？？ないとしたら、何というアホな仕様なのだろう。
- norm() が sparse matrix に対して使えなかったのが、使えるようになった (2022a → 2023b)。少しずつ改良されているのか。

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：Poisson 方程式に対する差分法, https://m-katsurada.sakura.ne.jp/labo/text/poisson.pdf (2000 年?〜).

[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〜).