

新 MATLAB 入門

桂田 祐史

2018 年 8 月 24 日, 2018 年 11 月 2 日

思い出すたびに MATLAB を採り上げて、にわか勉強で動くプログラムを書く、というのを続けてきて、ガラクタの山が残った (<http://nalab.mind.meiji.ac.jp/~mk/labo/text/matlab/>)。いいかげんに整理しよう (目的別にこうするのが良いだろう、というものをまとめる。結局は今回もすでに作ってあるものの寄せ集めであるが、一つの目的に複数のプログラムがあった場合は、なるべく一つにまとめる。)

使用上の注意: 「とにかく一つの文書にまとめておく」が主目的なので、全体の出来は非常に粗い。間違っているところも少なからずあると思われる。ちらっと見て間違っているように感じられても、確認する時間が取れないので、あえてそのまま放置してあるところが多い (ちら見で考えたことが正しいという保証はないので)。自分で再利用するときは、書いてあることを読んで、プログラムを動かしながら、内容を思い出し、確認してから先に進むつもりでいる。

目次

1	MATLAB とは何か	2
1.1	MATLAB とは、その起源	2
2	1 次元 Poisson 方程式	4
2.1	Dirichlet 境界値問題	4
2.2	Neumann 境界値問題	6
3	2 次元 Poisson 方程式	6
3.1	Dirichlet 境界値問題	6
3.2	Neumann 境界値問題	10
4	1 次元熱方程式	10
4.1	Dirichlet 境界条件	10
4.1.1	陽解法	10
4.1.2	陰解法	11
4.2	Neumann 境界条件	11
5	2 次元熱方程式	11
5.1	Dirichlet 境界条件	11
5.2	Neumann 境界条件	13
6	固有値問題	15

A	MATLAB メモ	15
A.1	基本	15
A.2	zeros(), ones(), eye(), rand()	16
A.3	行列とベクトルの基本	17
A.4	: に慣れる	19
A.5	添字ベクトル (?) の利用	20
A.6	lu()	22
A.7	diag()	25
A.8	kron()	26
A.9	linspace()	26
A.10	meshgrid()	27
A.11	meshgrid(m,n) と meshx(x,y,f)	29
A.12	misc: pause(), fprintf()	29
	A.12.1 pause()	29
	A.12.2 fprintf()	29
B	行列を作る	29
B.1	はじめに	29
B.2	Laplacian の近似	30
	B.2.1 1次元の場合	30
	B.2.2 2次元の場合	30
B.3	コピペして試す	32
C	INTLAB	34
C.1	いんところ	34
C.2	入手とインストール	34
C.3	簡単な使い方	35
D	row major vs. column major	35
D.1	MATLAB, Fortran は column major order	35
D.2	C は row major order	36
E	グラフィックス	37
F	固有値関係の命令	37

1 MATLAB とは何か

(あとまわし)

1.1 MATLAB とは、その起源

MATLAB (MATrix LABoratory) は、著名な線形演算ライブラリ LINPACK, EISPACK の開発でも中心的な役割を果たした Cleve Moler が、1980 年頃に製作したものが発展した数値実験環境 (「実験室」) である (当時の開発言語は FORTRAN)。彼は 1985 年に C 言語で MATLAB を書き直し、MathWorks 社を設立して販売を開始した (Moler は会長兼技師長であるとか — 現在もそうであるかは知らない)。

MATLAB の特徴

- インタープリター型言語である。そのため^a、
 - 対話的で使いやすいシステムになっている。
 - 注意深く利用しないと実行効率が低くなる^b(個々の命令の実行時に命令解釈のコストが必要なため、繰り返し処理を多用すると計算時間が長くなりがちである)。
- LAPACK などの各種数値計算ライブラリを内蔵している (これらのライブラリ群へのインターフェイスであると理解すべきかもしれない)。
- ベクトル、行列などのデータの型が始めから定義されているので、命令が簡潔になっていて、プログラミングも楽になった^c。

^aここで指摘することは、例えば Mathematica, Maple のような多くの数式処理系にも当てはまる。

^bここで述べたような注意は、かつてはパソコン上で BASIC 言語を使ってプログラムを開発する際の常識であったのだが、今ではあまり知られていないことなのだろう。

^cオブジェクト指向であり、データ構造が隠蔽されていると言って良いかもしれない。LAPACK などの利用で面倒な点の一つに、プログラマーにライブラリ中で定義されたデータ構造を正しくなぞったプログラムを書く努力が要求されるというものがあるが、MATLAB ではこれがなくなっている。この点は C++ で書かれたライブラリでも期待できることであるが。

MATLAB をいかに評価するか。筆者自身は最初は「ちょっと便利」くらいにしか感じなかったが、使い続けるうちに

案外大したものではないか

さらに

ひょっとするとコロンプスの卵で大発明？

と考えるようになった。このようなシステムを作るのは実は簡単で (実際、「真似」がたくさん出て来た)、しかし使ってみると分るが、非常に便利である。

日本の数学界ではあまり人気がない (というか知られていない) ようであるが、欧米や、日本でも工学の世界では浸透している。

MATLAB は改良が続けられていて、行列計算関係では、疎行列向きの処理法や反復法なども採り入れられている。偏微分方程式のシミュレーションへの応用にも「もってこい」レベルに成長した。

MATLAB を後を追ったシステムがたくさん開発されたが、MATLAB の言語仕様は「デファクト・スタンダード」となっている。以下 MATLAB と似たシステムをいくつか紹介しよう¹。いずれもソース・プログラム公開のフリーソフトウェアである。

(GNU) Octave MATLAB との互換性が高い。入門には十分であるし、用途を選べば実用性も高い。むしろ、中身が完全に公開されているということを積極的に評価すべきであるかもしれない。近年、GUI を採用したせいで、make が難しくなり、動作が不安定な時期があった (個人的には GUI に力を入れたことには、不満であった)。

Scilab MATLAB との互換性の程度は Octave よりも低いが、ソフトウェアとしての完成度はやや高いかもしれない (と感じた時期があった、現在でもそういう評価になるかは知らない)。

¹ <http://www.dspguru.com/sw/opensp/mathclo2.htm> などが参考になる。

2 1次元 Poisson 方程式

2.1 Dirichlet 境界値問題

$$-u''(x) = f(x) \quad (x \in (0, L)), \quad u(0) = u(L) = 0$$

$$h = \frac{L}{N}, \quad x_i = ih \quad (i = 0, 1, \dots, N).$$

$$\frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \end{pmatrix}.$$

```
laplacian1d.m
```

```
% -(d/dx)^2 の差分近似
function a=laplacian1d(m)
    e=ones(m,1);
    a=spdiags([-e 2*e -e],[-1:1,m,m]); % 疎行列なので圧縮した形式
end
```

```
>> a=laplacian1d(3);
>> full(a)
```

```
ans =
     2     -1     0
    -1     2    -1
     0    -1     2
```

$I = (0, L)$ を N 等分する差分近似の場合は

```
>> L=1;
>> n=5;
>> h=L/n;
>> a=laplacian1d(n-1)/(h*h);
>> full(a)
```

```
ans =

    50.0000   -25.0000         0         0
   -25.0000    50.0000   -25.0000         0
         0   -25.0000    50.0000  -25.0000
         0         0   -25.0000    50.0000
```

$I = (0, L)$ における

$$-u''(x) = 1, \quad u(0) = u(L) = 0$$

を解くには ($f \equiv 1$ とするのは安直だけど)

```
poisson1d_test1.m
L=1
n=10
h=L/n
x=linspace(0,L,n+1);
a=laplacian1d(n-1)/(h*h);
f=ones(n-1,1);
u=a\f;
u=[0; u; 0];
plot(x,u)
figure(gcf)
```

とする。

$f(x) = \sin x$ の場合は

```
poisson1d_test2.m
L=1
n=10
h=L/n
x=linspace(0,L,n+1);
a=laplacian1d(n-1)/(h*h);
f=sin(x(2:n))';
u=a\f;
u=[0; u; 0];
plot(x,u)
figure(gcf)
```

とする。

$f(x) = x(1-x)$ の場合は

```
poisson1d_test3.m
L=1
n=10
h=L/n
x=linspace(0,L,n+1);
a=laplacian1d(n-1)/(h*h);
f=x.*(1-x);
f=f(2:n)';
u=a\f;
u=[0; u; 0];
plot(x,u)
figure(gcf)
```

とする。

2.2 Neumann 境界値問題

(場所取り)

3 2次元 Poisson 方程式

3.1 Dirichlet 境界値問題

長方形領域 $\Omega = (a, b) \times (c, d)$ における Poisson 方程式の境界値問題

$$-\Delta u = f \quad \text{in } \Omega, \quad u = 0 \quad (\text{on } \partial\Omega)$$

を考える。

$$\begin{aligned} h_x &= \frac{b-a}{N_x}, & h_y &= \frac{d-c}{N_y}, \\ x_i &= a + ih_x \quad (i = 0, 1, \dots, N_x), & y_j &= c + jh_y \quad (j = 0, 1, \dots, N_y), \\ u_{ij} &= u(x_i, y_j) \quad (i = 0, 1, \dots, N_x; j = 0, 1, \dots, N_y) \end{aligned}$$

とおき、 u_{ij} の近似値を U_{ij} を求めるための差分方程式は

$$(1) \quad - \left(\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h_x^2} + \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h_y^2} \right) = f(x_i, y_j)$$
$$(1 \leq i \leq N_x - 1, 1 \leq j \leq N_y - 1),$$

$$(2) \quad U_{0,j} = U_{N_x,j} = 0 \quad (0 \leq j \leq N_y),$$

$$(3) \quad U_{i,0} = U_{i,N_y} = 0 \quad (0 \leq i \leq N_x)$$

Dirichlet 境界値問題については、 U_{ij} ($1 \leq i \leq N_x, 1 \leq j \leq N_y$) が未知数になる。桂田 [1] というノートでは

$$(4) \quad U_\ell = U_{i,j}, \quad \ell = i + j(N_x - 1) \quad (1 \leq i \leq N_x - 1, 1 \leq j \leq N_y - 1)$$

とおき、 $\mathbf{U} = \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_N \end{pmatrix}$ を未知ベクトルとする連立1次方程式を考えた。連立1次方程式の係数

行列は

$$A = \frac{1}{h_y^2}(2I_{N_y-1} - J_{N_y-1}) \otimes I_{N_x-1} + I_{N_y-1} \otimes \frac{1}{h_x^2}(2I_{N_x-1} - J_{N_x-1})$$

であった。

以下のプログラムでは (4) の代わりに

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

とおく。この場合の係数行列は

$$A = I_{N_x-1} \otimes \frac{1}{h_y^2}(2I_{N_y-1} - J_{N_y-1}) + \frac{1}{h_x^2}(2I_{N_x-1} - J_{N_x-1}) \otimes I_{N_y-1}.$$

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 Fortran 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 次元の番号付けする
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));
```

poisson2d.m

```
% 長方形領域 (0,W) × (0,H) で Poisson 方程式の同次 Dirichlet 境界値問題を解く
W=3.0;
H=2.0;
nx=30;
ny=20;
m=nx-1;
n=ny-1;
% 連立方程式を作成して解く
% MATLAB の行列は Fortran と同様の column first であり、
% 「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 次元の番号付けする
A=poisson_coef(W, H, nx, ny);
%
x=linspace(0,W,nx+1); % x=[x_0,x_1,...,x_nx]
y=linspace(0,H,ny+1); % y=[y_0,y_1,...,y_ny]
[X,Y]=meshgrid(x,y);
% f ≡ 1 の場合
%F=ones(m*n,1);
f=-2*(X.^2-3*X+Y.^2-2*Y);
f=f(2:ny,2:nx);
F=f(:);
%
U=zeros(n,m);
U(:)=A\F;
% 境界値 0 をつける
u=zeros(ny+1,nx+1);
u(2:ny,2:nx)=U;
%
% グラフの鳥瞰図
clf
colormap hsv
subplot(1,2,1);
mesh(X,Y,u);
colorbar
% 等高線
subplot(1,2,2);
contour(X,Y,u);
%
disp(' 図を保存する');
print -dpdf poisson2d.pdf % 利用できるフォーマットは doc print で分かる
```

```
print -dpng poisson2d.png % 利用できるフォーマットは doc print で分かる
print -deps poisson2d.eps % 利用できるフォーマットは doc print で分かる
```

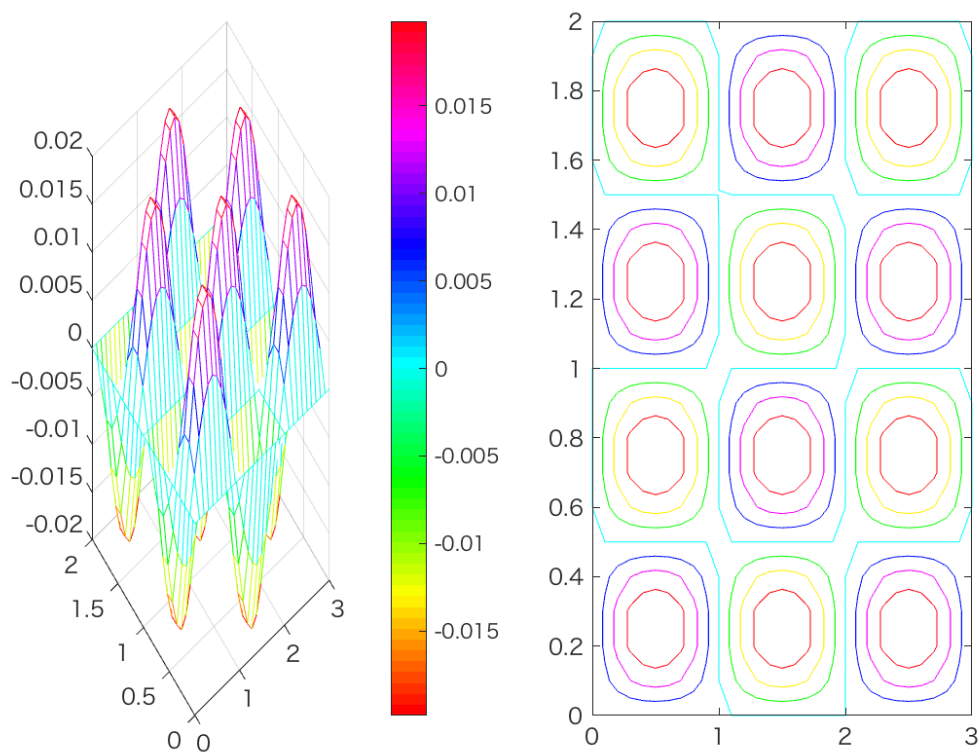


図 1: poisson2d.m の結果

まったく別の時期に作ったプログラム。ほとんど同じで我ながら唾然とする。最初のプログラムが `mesh()` と `contour()` で、鳥瞰図と等高線を別々に描いたが、こちらは `meshc()` で同時に描いている。

poisson2d.v2.m

```
% poisson2d.m --- Poisson equation  $-\Delta u = \sin(\pi x)\sin(2\pi y)$  ( $0 < x < 3$ ,  $0 < y < 2$ ),  $u=0$   
%  $[0,3] \times [0,2]$  を  $30 \times 20$  に分割する  
a=0; b=3; c=0; d=2;  
nx=30; ny=20;  
%nx=6; ny=4;  
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(pi*x).*sin(2*pi*y);  
% 係数行列  
hx=(b-a)/nx; hy=(d-c)/ny;  
e=ones(nx-1,1);  
ax=spdiags([-e 2*e -e],-1:1,nx-1,nx-1)/(hx*hx);  
e=ones(ny-1,1);  
ay=spdiags([-e 2*e -e],-1:1,ny-1,ny-1)/(hy*hy);  
a=kron(speye(nx-1),ay)+kron(ax,speye(ny-1));  
% F の境界部分の値を削除して、1次元化  
f=F(2:end-1,2:end-1);  
f=f(:);  
%  
u=zeros(ny+1,nx+1);  
u(2:end-1,2:end-1)=reshape(a\f,ny-1,nx-1);  
%  
figure('Name','Poisson equation  $-\Delta u = \sin(\pi x)\sin(2\pi y)$  ( $0 < x < 3$ ,  $0 < y < 2$ )')  
meshc(x,y,u)  
figure(gcf)
```

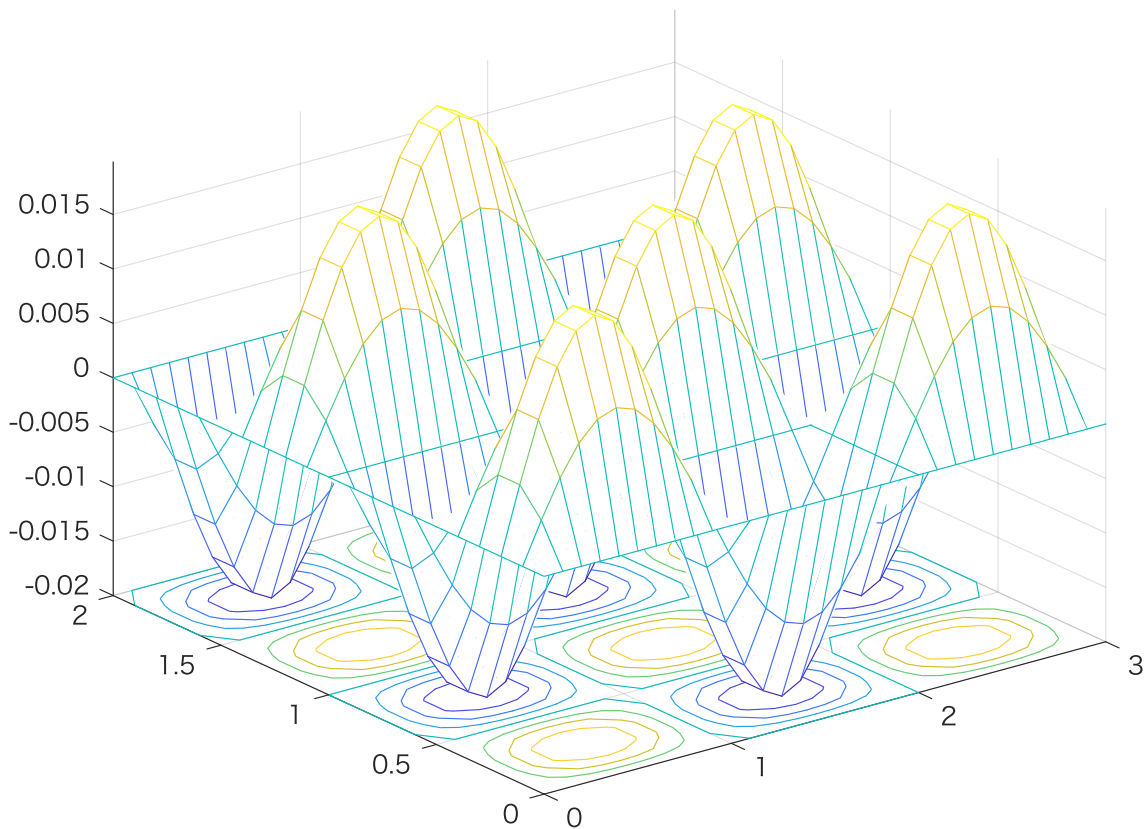


図 2: poisson2d.v2.m の結果

3.2 Neumann 境界値問題

4 1次元熱方程式

4.1 Dirichlet 境界条件

4.1.1 陽解法

```
% heat1d_e.m -- 空間 1 次元熱方程式, 同次 Dirichlet 境界条件

% 区間
a=0
b=1
% N 等分
N=100
% N 等分点
x=linspace(a,b,N+1);
% 初期値 (実は u=min(x,1-x); ですむ)
for i=0:N
    if x(i+1)<0.5
        u(i+1)=x(i+1);
    else
        u(i+1)=1-x(i+1);
    end
end
% 初期値のグラフを描いて 1 秒待つ
%fig=figure % 新しく図のウィンドウを出すかどうか
plot(x,u)
fig(gcf; figure(fig) % こうすると visible になる
axis([0 1 -0.1 1.1])
title('heat equation')
pause(1)

% 差分法の準備
h=(b-a)/N
lambda=0.5
tau=lambda*h*h
tMax=1
nMax=tMax/tau
newu=zeros(1,N+1);
% t が dt 増えるごとに描画
dt = 0.01;
nskip = round(dt / tau);
for n=1:nMax
    newu(2:N)=(1-2*lambda)*u(2:N)+lambda*(u(1:N-1)+u(3:N+1));
    newu(1)=0;
    newu(N+1)=0;
    if mod(n,nskip)==0
        plot(x,newu)
        axis([0 1 -0.1 1.1])
        title(['heat equation, t=', num2str(n*tau, '%4.2f')])
        pause(0.1);
    end
    u=newu;
end
end
```

4.1.2 陰解法

```
% heat1d_i.m -- 空間1次元熱方程式, 同次 Dirichlet 境界条件

% 区間
a=0;
b=1;
% N 等分
N=100;
% N 等分点
x=linspace(a,b,N+1);
% 初期値
u=min(x,1-x);
% 初期値のグラフを描いて1秒待つ
%fig=figure
plot(x,u);
fig(gcf; figure(fig) % こうすると visible になる
axis([0 1 -0.1 1.1]);
title('heat equation');
pause(1);

tMax=1;
h=(b-a)/N;
lambda=0.5;
tau=lambda*h*h;
theta=0.5; % Crank-Nicolson
a=sparse((1+2*theta*lambda)*eye(N-1,N-1)-theta*lambda*(diag(ones(N-2,1),1)+diag(ones(N-2,1),-1)));

nMax=tMax/tau;
% t が dt 増えるごとに描画
dt = 0.01;
nskip = round(dt / tau);
for n=1:nMax
    f=(1-2*(1-theta)*lambda)*u(2:N)+(1-theta)*lambda*(u(1:N-1)+u(3:N+1));
    u(2:N)=a\f';
    u(1)=0;
    u(N+1)=0;
    if mod(n,nskip)==0
        plot(x,u);
        axis([0 1 -0.1 1.1]);
        title(['heat equation, t=', num2str(tau*n, '%4.2f')])
        pause(0.1);
    end
end
end
```

4.2 Neumann 境界条件

5 2次元熱方程式

5.1 Dirichlet 境界条件

「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」²

²<http://nalab.mind.meiji.ac.jp/~mk/labo/text/heat2d.pdf>

heat2d.m

```
% 長方形領域における熱方程式  $u_t = \Delta u$  (Dirichlet 境界条件) を解くための差分方程式
%  $A U^{n+1} = B U^n$ 
% の行列 A, B を求める。(2015/5/1 作成, 2015/5/30 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(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 heat2d
    a=0; b=2; c=0; d=1;
    Nx=100;
    Ny=50;
    hx=(b-a)/Nx;
    hy=(d-c)/Ny;
    theta=0.5;
    tau=0.5/(1/hx^2+1/hy^2);
    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);
if ((Nx <= 5) && (Ny <= 5))
    A
    B
end
% 格子点の座標ベクトル  $x=(x_1,x_2,\dots,x_{Nx+1})$ ,  $y=(y_1,y_2,\dots,y_{Ny+1})$ 
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);
%
% 初期値のグラフを描く
disp('初期値')
mesh(x,y,u);
[AL,AU,ap]=lu(A,'vector');
Tmax=1;
t=tau;
disp('繰り返し')
k=0;
dt=0.005;
skip=dt/tau;
U=reshape(u(2:Ny,2:Nx),(Nx-1)*(Ny-1),1);
while t<=Tmax
    U=B*U;
    U=AU\ (AL\U(ap,:));
    if mod(k,skip)==0
        u(2:Ny,2:Nx)=reshape(U,Ny-1,Nx-1);
        meshc(x,y,u);
        axis([a b c d -1 1]);
        drawnow;
    end
    t=t+tau
    k=k+1;
end
end
```

heat2d_mat.m

```
% 長方形領域における熱方程式  $u_t = \Delta u$  (Dirichlet 境界条件) を解くための差分方程式
%  $A U^{n+1} = B U^n$ 
% 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 changes 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);
```

5.2 Neumann 境界条件

「Neumann 境界条件下の熱方程式に対する差分法」³

heat2n.m

```
% 長方形領域における熱方程式  $u_t = \Delta u$  (Dirichlet 境界条件) を解くための差分方程式
%  $A U^{n+1} = B U^n$ 
% の行列 A, B を求める。
% 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(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 heat2d
    a=0; b=2; c=0; d=1;
    Nx=100;
    Ny=50;
    hx=(b-a)/Nx;
```

³<http://nalab.mind.meiji.ac.jp/~mk/labo/text/heat2n.pdf>

```

hy=(d-c)/Ny;
theta=0.5;
tau=0.5/(1/hx^2+1/hy^2);
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]=heat2n_mat(Nx,Ny,lambdax,lambday,theta);
if ((Nx <= 5) && (Ny <= 5))
    A
    B
end
% 格子点の座標ベクトル  $x=(x_1,x_2,\dots,x_{Nx+1})$ ,  $y=(y_1,y_2,\dots,y_{Ny+1})$ 
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);
if Nx<=5 && Ny<=5
    x
    y
    X
    Y
    u
end
%
% 初期値のグラフを描く
disp('初期値')
mesh(x,y,u);
[AL,AU,AP]=lu(A);
Tmax=1;
t=tau;
disp('繰り返し')
k=0;
dt=0.005;
skip=dt/tau;
v=u;
v(1,:)=v(1,+)/sqrt(2); v(Ny+1,:)=v(Ny+1,+)/sqrt(2);
v(:,1)=v(:,1)/sqrt(2); v(:,Nx+1)=v(:,Nx+1)/sqrt(2);
V=reshape(v,(Nx+1)*(Ny+1),1);
while t<=Tmax
    V=AU\ (AL\ (AP*(B*V)));
    if mod(k,skip)==0
        u(1:Ny+1,1:Nx+1)=reshape(V,Ny+1,Nx+1);
        u(1,:)=u(1,)*sqrt(2); u(Ny+1,:)=u(Ny+1,)*sqrt(2);
        u(:,1)=u(:,1)*sqrt(2); u(:,Nx+1)=u(:,Nx+1)*sqrt(2);
        meshc(x,y,u);
        axis([a b c d -1 1]);
        drawnow;
    end
    t=t+tau
    k=k+1;
end
end

```

```
heat2n.mat.m
```

```
% 長方形領域における熱方程式  $u_t = \Delta u$  (Dirichlet 境界条件) を解くための差分方程式
%  $A U^{n+1} = B U^n$ 
% の行列 A, B を求める。
% 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;
% heat2n_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,lambda_x,lambda_y,theta)
    Ix=speye(Nx+1,Nx+1);
    Iy=speye(Ny+1,Ny+1);
    vx=[sqrt(2); ones(Nx-2,1); sqrt(2)];
    Jx=sparse(diag(vx,1)+diag(vx,-1));
    vy=[sqrt(2); ones(Ny-2,1); sqrt(2)];
    Jy=sparse(diag(vy,1)+diag(vy,-1));
    Kx=2*Ix-Jx;
    Ky=2*Iy-Jy;
% column first
    A=kron(Ix,Iy)+theta*lambda_y*kron(Ix,Ky)+theta*lambda_x*kron(Kx,Iy);
    B=kron(Ix,Iy)-(1-theta)*lambda_y*kron(Ix,Ky)-(1-theta)*lambda_x*kron(Kx,Iy);
% row first
%   A=kron(Iy,Ix)+theta*lambda_x*kron(Iy,Kx)+theta*lambda_y*kron(Ky,Ix);
%   B=kron(Iy,Ix)-(1-theta)*lambda_x*kron(Iy,Kx)-(1-theta)*lambda_y*kron(Ky,Ix);
```

6 固有値問題

A MATLABメモ

A.1 基本

- コマンドウィンドウのプロンプト `>>` に対してコマンドを入力する。

```
>> 1+1

ans = 2
```

- 行単位の編集機能がある。コマンド履歴 (ヒストリー) があり、呼び出せる。タブ入力により補完が出来る。
- `doc` 名前 で MATLAB 組み込みの関数や変数についてのヘルプを呼び出せる。
- `変数名=式` で変数に値を代入し、表示する。`変数名=式;` とコロンをつけると変数に値を代入するが、値は表示しない。

```
>> a=1+1

a =
    2

>> a=1+1;
>> a
a =
    2
```

- `who` で使っている変数名一覧を表示する。`whos` で使っている変数の情報(型、サイズ)一覧を表示する。
- `clear` ですべての変数を消去する。`clear` 変数名 で指定した変数を消去する。

```
>> a=1+1

a =
    2

>> who
変数:

a

>> whos
Name      Size      Bytes  Class  Attributes

a         1x1         8  double

>> clear
>> whos
>>
```

- プログラム・ファイルの編集は(テキスト・エディターを使うことも出来るが) `edit` 関数名 あるいは `edit` スクリプト名
- パーセント記号 `%` から行末までは注釈になる。
- 行を継続したいときは行末に継続用の記号 `...` (3つ以上の連続ピリオド)をおく。

A.2 `zeros()`, `ones()`, `eye()`, `rand()`

- `zeros(m,n)`, `ones(m,n)`, `eye(m,n)`, `rand(m,n)` はそれぞれ、 m 行 n 列の零行列、全ての成分が1の行列、対角成分が1でそれ以外が0の行列、乱数行列を返す。`zeros(n)` は `zeros(n,n)` と同じ (`ones()`, `eye()`, `rand()` も同様)。
- `'` は行列・ベクトルのエルミート転置、`.'` は転置を表す。 x と y が同じ次元のたてベクトルであるとき、 $y'*x$ は内積に等しい (`dot(x,y)` と同じ)。

A.3 行列とベクトルの基本

カンマ , と空白は、ともに次の列に移ることを意味する。
セミコロン ; と改行は、ともに次の行に移ることを意味する。

```
>> [1 2 3]

ans =
     1     2     3

>> [1,2,3]

ans =
     1     2     3

>> [1;2;3]

ans =
     1
     2
     3

>> [1
2
3]

ans =
     1
     2
     3

>> [1,2;3,4]

ans =
     1     2
     3     4

>> >> [1,2
3,4
5,6]

ans =
     1     2
     3     4
     5     6

>>
```

- 配列のサイズは `size()` で求まる。

```

>> x=1:2

x =
    1    2    3

>> size(x)

ans =
    1    3

>> [m,n]=size(x)

m =
    1

n =
    3

>> a=rand(3,2)

a =
    0.9649    0.9572
    0.1576    0.4854
    0.9706    0.8003

>> [m,n]=size(a)

m =
    3

n =
    2

```

- 配列の添字は 1 から始まる。
- 1次元配列 a の第 i 成分は $a(i)$ で表す。
- 2次元配列 A の第 (i, j) 成分は $A(i, j)$ で表す。

行列とベクトル よく使う機能早見表

<code>[m,n]=size(a)</code>	行列 a のサイズ (行の数、列の数)
<code>eye(m,n)</code>	m 行 n 列の単位行列 ((i, j) 成分が δ_{ij})
<code>zeros(m,n)</code>	m 行 n 列の零行列
<code>ones(m,n)</code>	(m 行 n 列の) 成分がすべて 1 の行列
<code>rand(m,n)</code>	(m 行 n 列の) 成分が乱数の行列
<code>diag()</code>	対角行列 (を少しずらした行列)
<code>a'</code>	a の Hermite 共役 (実行列、実ベクトルの場合転置)
<code>a.'</code>	a の転置
<code>det(a)</code>	a の行列式
<code>inv(a)</code>	a の逆行列 (大きいときは使用を控えるべき)
<code>y'*x</code>	縦ベクトル x, y の内積
<code>norm(x)</code>	x のノルム (成分の絶対値の二乗の和の平方根)
<code>tril(a)</code>	a の下三角部分
<code>triu(a)</code>	a の上三角部分
<code>a(i,:)</code>	a の第 i 行ベクトル
<code>a(:,j)</code>	a の第 j 列ベクトル
<code>a(i1:i2,j1:j2)</code>	a の第 $i1 \sim i2$ 行、第 $j1 \sim j2$ 列の部分のブロック

A.4 : に慣れる

```
>> 1:10

ans =
     1     2     3     4     5     6     7     8     9    10

>> 0:0.2:1

ans =
     0    0.2000    0.4000    0.6000    0.8000    1.0000
```

$x_0 \leq x_1$ とするとき、 $x_0:x_1$ は、 n を $x_0 + n \leq x_1, x_0 + n + 1 > x_1$ となる整数として、 $[x_0, x_0 + 1, \dots, x_0 + n]$ という横ベクトルを返す。

$x_0 \leq x_1, \Delta x > 0$ とするとき、 $x_0:\Delta x:x_1$ は、 n を $x_0 + n\Delta x \leq x_1, x_0 + (n+1)\Delta x > x_1$ となる整数として、 $[x_0, x_0 + \Delta x, x_0 + 2\Delta x, \dots, x_0 + n\Delta x]$ という横ベクトルを返す。

$x_0 \geq x_1, \Delta x < 0$ とするとき、 $x_0:\Delta x:x_1$ は、 n を $x_0 + n\Delta x \geq x_1, x_0 + (n+1)\Delta x < x_1$ となる整数として、 $[x_0, x_0 + \Delta x, x_0 + 2\Delta x, \dots, x_0 + n\Delta x]$ という横ベクトルを返す。

行列 a に対して、 $a(:)$ は 1次元化したベクトルを返す。

```

>> a=[1,2;3,4]

a =
     1     2
     3     4

>> v=a(:)

v =
     1
     3
     2
     4

>> b=reshape(v,2,2)

b =
     1     2
     3     4

```

A.5 添字ベクトル(?)の利用

ベクトル、行列に対して、添字の代わりに“添字ベクトル”を指定することで、色々なことが出来る。

- ブロックの指定。例えば行列 A の 3～5 行、4～8 列からなるブロックは、 $A(3:5,4:8)$ で表せる。

```

>> a=(1:9)'+*(1:9)

a =
     1     2     3     4     5     6     7     8     9
     2     4     6     8    10    12    14    16    18
     3     6     9    12    15    18    21    24    27
     4     8    12    16    20    24    28    32    36
     5    10    15    20    25    30    35    40    45
     6    12    18    24    30    36    42    48    54
     7    14    21    28    35    42    49    56    63
     8    16    24    32    40    48    56    64    72
     9    18    27    36    45    54    63    72    81

>> a(3:5,4:8)

ans =
    12    15    18    21    24
    16    20    24    28    32
    20    25    30    35    40

>>

```

- ブロックの特別な場合として、第 i 行ベクトルは $A(i,:)$ 、第 j 列ベクトルは $A(:,j)$

```

>> a=rand(4,3)

a =
    0.1419    0.9595    0.9340
    0.4218    0.6557    0.6787
    0.9157    0.0357    0.7577
    0.7922    0.8491    0.7431

>> a(2,:)

ans =
    0.4218    0.6557    0.6787

>> a(:,3)

ans =
    0.9340
    0.6787
    0.7577
    0.7431

>>

```

- 行の置換、列の置換なども行える。

```

>> a(9:-1:1,:)

ans =
     9     18     27     36     45     54     63     72     81
     8     16     24     32     40     48     56     64     72
     7     14     21     28     35     42     49     56     63
     6     12     18     24     30     36     42     48     54
     5     10     15     20     25     30     35     40     45
     4      8     12     16     20     24     28     32     36
     3      6      9     12     15     18     21     24     27
     2      4      6      8     10     12     14     16     18
     1      2      3      4      5      6      7      8      9

>> a([1 3 2 6 5 4 9 8 7],:)

ans =
     1      2      3      4      5      6      7      8      9
     3      6      9     12     15     18     21     24     27
     2      4      6      8     10     12     14     16     18
     6     12     18     24     30     36     42     48     54
     5     10     15     20     25     30     35     40     45
     4      8     12     16     20     24     28     32     36
     9     18     27     36     45     54     63     72     81
     8     16     24     32     40     48     56     64     72
     7     14     21     28     35     42     49     56     63

```

- 配列の最後の添字は `end` で表せる。例えばベクトル `x` の最初と最後の成分を除くには `x=x(2:end-1)` とすれば良い。

```

>> x=1:10

x =

     1     2     3     4     5     6     7     8     9    10

>> x(2:end-1)

ans =

     2     3     4     5     6     7     8     9

>> x=(1:10)′

x =

     1
     2
     3
     4
     5
     6
     7
     8
     9
    10

>> x=x(2:end-1)

x =

     2
     3
     4
     5
     6
     7
     8
     9

>>

```

A.6 lu()

A が与えられたとき、 $LU = PA$ を満たす置換行列 P , 下三角行列 L , 上三角行列 U を求めることを A を LU 分解すると言う。

```

a=hilb(4)

とすると

a =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429

```

```
[L U P]=lu(a)
```

とすると

```
L =
  1.0000    0    0    0
  0.3333    1.0000    0    0
  0.5000    1.0000    1.0000    0
  0.2500    0.9000   -0.6000    1.0000
```

```
U =
  1.0000    0.5000    0.3333    0.2500
  0    0.0833    0.0889    0.0833
  0    0    -0.0056   -0.0083
  0    0    0    0.0004
```

```
P =
  1    0    0    0
  0    0    1    0
  0    1    0    0
  0    0    0    1
```

念のため

```
norm(L*U-P*a)
```

とすると

```
ans =
  3.9252e-17
```

$[l \ u]=lu(a)$ とすると 置換行列を省いて $[l \ u]=lu(a)$ とすると、 l には PL が、 u には U が代入される。

```
[L U P]=lu(a);
[L0 U0]=lu(a)
```

とすると

```
L0 =
  1.0000    0    0    0
  0.5000    1.0000    1.0000    0
  0.3333    1.0000    0    0
  0.2500    0.9000   -0.6000    1.0000
```

```
U0 =
  1.0000    0.5000    0.3333    0.2500
  0    0.0833    0.0889    0.0833
  0    0    -0.0056   -0.0083
  0    0    0    0.0004
```

本当に $L0 = PL$, $U0 = U$ かチェックする。

```
norm(U0-U)
```

```
norm(L0-P*L)
```

どちらも0になる。

```
[L U p]=lu(a,'vector')
```

とすると

```
L =  
 1.0000         0         0         0  
 0.3333    1.0000         0         0  
 0.5000    1.0000    1.0000         0  
 0.2500    0.9000   -0.6000    1.0000
```

```
U =  
 1.0000    0.5000    0.3333    0.2500  
 0         0.0833    0.0889    0.0833  
 0         0        -0.0056   -0.0083  
 0         0         0         0.0004
```

```
p =  
 1     3     2     4
```

p は置換行列 P の “情報を持った” ベクトルである。

- ベクトル c に対して、 Pc を求めるには $P*c$ とする代わりに、 $c(p)$ とすれば良い。
- 行列 C に対して、 PC を求めるには $P*C$ とする代わりに、 $C(p,:)$ とすれば良い。

$PA = LU$ となっているとき、

$$Ax = b \Leftrightarrow PAx = Pb \Leftrightarrow LUx = Pb$$

であるから、 x を求めるには、 $U \setminus (L \setminus (P*b))$ 、あるいは $U \setminus (L \setminus b(p))$ とする。

解が分かりやすい連立1次方程式を用意しよう。

```
x=(1:4)';  
b=a*x;
```

解は、もちろん

```
a\b
```

とすれば求まるが、 $[L U P]=lu(a)$; で求めた L, U, P を使うには

```
U \ (L \ (P*b))
```

とすれば良く、 $[L U p]=lu(a,'vector')$; で求めた L, U, p を使うには

```
U \ (L \ b(p))
```

とすれば良い。


```

>> x=(1:4)';

x =
     1
     2
     3
     4

>> b=a*x;
>> a\b

ans =
     1.0000
     2.0000
     3.0000
     4.0000

>> U\(L\(P*b))

ans =
     1.0000
     2.0000
     3.0000
     4.0000

>> U\(L\b(p))

ans =
     1.0000
     2.0000
     3.0000
     4.0000

```

A.7 diag()

- `diag`(ベクトル) とすると、ベクトルを対角成分に埋め込んだ正方行列を返す。

```

>> diag(1:5)

ans =
     1     0     0     0     0
     0     2     0     0     0
     0     0     3     0     0
     0     0     0     4     0
     0     0     0     0     5

```

- `diag`(行列) とすると、行列の対角成分からなるベクトルを返す。

```
>> a=(1:9)'+*(1:9);
>> diag(a)

ans =
     1
     4
     9
    16
    25
    36
    49
    64
    81
```

ゆえに $\text{diag}(\text{diag}(\text{行列}))$ とすると、行列の対角成分以外を0クリアした対角行列が得られる。

```
>> a=(1:9)'+*(1:9);
>> diag(diag(a))

ans =
     1     0     0     0     0     0     0     0     0
     0     4     0     0     0     0     0     0     0
     0     0     9     0     0     0     0     0     0
     0     0     0    16     0     0     0     0     0
     0     0     0     0    25     0     0     0     0
     0     0     0     0     0    36     0     0     0
     0     0     0     0     0     0    49     0     0
     0     0     0     0     0     0     0    64     0
     0     0     0     0     0     0     0     0    81
```

A.8 kron()

行列 $A \in \mathbb{C}^{k \times \ell}$, $B \in \mathbb{C}^{m \times n}$ の Kronecker 積 $A \otimes B$ は

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1\ell}B \\ \vdots & \ddots & \vdots \\ a_{k1}B & \cdots & a_{k\ell}B \end{pmatrix} \quad (\text{ブロック行列の記法})$$

で定義される。

$\text{kron}(a,b)$ で $A \otimes B$ が計算できる。

A.9 linspace()

$\text{linspace}(a,b,m)$ は、 $[a,b]$ を $m-1$ 等分した点 (全部で m 個) の座標を並べた横ベクトルを返す。

n 等分点が欲しければ $\text{linspace}(a,b,n+1)$ とする。

A.10 meshgrid()

```
>> nx=5; ny=3;
>> X=linspace(0,1,nx+1)

X =
     0     0.2000     0.4000     0.6000     0.8000     1.0000

>> Y=linspace(2,3,ny+1)

Y =
     2.0000     2.3333     2.6667     3.0000

>> [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

y =
     2.0000     2.0000     2.0000     2.0000     2.0000     2.0000
     2.3333     2.3333     2.3333     2.3333     2.3333     2.3333
     2.6667     2.6667     2.6667     2.6667     2.6667     2.6667
     3.0000     3.0000     3.0000     3.0000     3.0000     3.0000

>> whos
  Name      Size      Bytes  Class  Attributes
  ---      -
  X         1x6         48  double
  Y         1x4         32  double
  nx        1x1          8  double
  ny        1x1          8  double
  x         4x6        192  double
  y         4x6        192  double
```

$X = [x_1, x_2, \dots, x_{N_x+1}] \in \mathbb{R}^{N_x+1}$, $Y = [y_1, y_2, \dots, y_{N_y+1}] \in \mathbb{R}^{N_y+1}$ とするとき、 $[x,y]=\text{meshgrid}(X,Y)$ とすると、 x と y は $\mathbb{R}^{(N_y+1) \times (N_x+1)}$ の要素となる。 $x(j,i)$ は x_i , $y(j,i)$ は y_j である。

MATLAB の 2次元配列は column first で並んでいるので、例えば x の成分は、メモリー内で、

```
x(1,1), x(2,1), x(3,1), x(4,1),
x(1,2), x(2,2), x(3,2), x(4,2),
...
x(1,5), x(2,5), x(3,5), x(4,5),
x(1,6), x(2,6), x(3,6), x(4,6)
```

と並んでいる。

```

>> x(:)
ans =
    0
    0
    0
    0
    0.2000
    0.2000
    0.2000
    0.2000
    0.4000
    0.4000
    0.4000
    0.4000
    0.6000
    0.6000
    0.6000
    0.6000
    0.8000
    0.8000
    0.8000
    0.8000
    1.0000
    1.0000
    1.0000
    1.0000

>> y(:)'
ans =
  1 列から 9 列
  2.0000    2.3333    2.6667    3.0000    2.0000    2.3333    2.6667    3.0000    2.0000

 10 列から 18 列
 2.3333    2.6667    3.0000    2.0000    2.3333    2.6667    3.0000    2.0000    2.3333

 19 列から 24 列
 2.6667    3.0000    2.0000    2.3333    2.6667    3.0000

>>

```

A.11 meshgrid(m,n) と meshx(x,y,f)

```
testofmeshgrid.m
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)
```

A.12 misc: pause(), fprintf()

A.12.1 pause()

指定した秒数だけ待ってくれる C 言語の `sleep()` が欲しい、と考えたが、MATLAB では `pause()` という関数がそれをしてくれる。

A.12.2 fprintf()

`disp()` で表示するのは大雑把すぎる。C 言語の `printf()` はないけれど、`fprintf()` というのがある。

```
fprintf('n=%d, t=%5.2f\n', n, t);
```

のようなことが出来る。C プログラマーには説明が不要だろう (`printf("n=%d, t=%5.2f\n", n, t);` と同じことをする)。

B 行列を作る

B.1 はじめに

どんな行列 $A = (a_{ij}) \in \mathbb{R}^{m \times n}$ も、成分 a_{ij} を表す式が分かれば

```
a=zeros(m,n);
for i=1:m
    for j=1:n
        a(i,j)=(i,j 成分を表す式);
    end
end
```

で作成出来る (最初に `zeros()` を使うのが一つの定跡的なテクニック)。

しかし、疎行列の場合など、二重ループは出来れば避けたい。

B.2 Laplacian の近似

本文中に書いたことで再掲ということになる (本文の方は問題に重点があり、こちらは文法や MATLAB の命令の方に重点がある)。

以下、Dirichlet 境界条件の場合のコードを中心に示す。

B.2.1 1次元の場合

$$A = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N-1, N-1}, \quad h = \frac{L}{N}$$

は頻出する。

```
h=L/N;  
e=ones(N-1,1);  
a=spdiags([-e 2*e -e],-1:1,N-1,N-1)/(h*h);
```

ずっと以前は、疎行列関係の命令を知らずに次のように作っていた。

```
h=L/N;  
e=ones(N-1,1);  
a=(2*eye(N-1,N-1)-diag(e,1)-diag(e,-1))/(h*h);
```

(この後で、遅ればせながら `a=sparse(a)`; としてメモリを節約するという手はあるかも。)

B.2.2 2次元の場合

長方形領域 $\Omega = (0, W) \times (0, H)$ を横方向に N_x 等分、縦方向に N_y 等分する。

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

$$x_i = ih_x \quad (i = 0, 1, \dots, N_x),$$

$$y_j = jh_y \quad (j = 0, 1, \dots, N_y),$$

$$u_{ij} = u(x_i, y_j) \quad (i = 0, 1, \dots, N_x; j = 0, 1, \dots, N_y).$$

u_{ij} の近似値 U_{ij} を求めることが目標となることが多い。

例えば Poisson 方程式の Dirichlet 境界値問題

$$-\Delta u = f \quad \text{in } \Omega, \quad u = \phi \quad \text{on } \partial\Omega$$

を解く場合、 U_{ij} ($1 \leq i \leq N_x - 1, 1 \leq j \leq N_y - 1$) が未知数となる。

$$m := N_x - 1, \quad n := N_y - 1$$

とおく。

連立1次方程式を行列とベクトルで表示するためには、 U_{ij} を1次元的に並べたベクトルを作る必要がある。例えば

$$\mathbf{U} = (U_{1,1}, U_{2,1}, \dots, U_{m,1}, U_{1,2}, U_{2,2}, \dots, U_{m,2}, U_{1,3}, U_{2,3}, \dots, U_{m,3}, \dots, U_{1,n}, U_{2,n}, \dots, U_{m,n})^T.$$

\mathbf{U} の成分を U_ℓ と書くとき

$$U_{i,j} = U_{i+n(j-1)}$$

という式が成り立つことが分かる。

$$(6) \quad \ell = i + n(j - 1)$$

とおくと

$$\ell - 1 = (i - 1) + n(j - 1), \quad 0 \leq i - 1 \leq n - 1$$

が成り立つので、 $i - 1$ は $\ell - 1$ を n で割った余り、 $j - 1$ は $\ell - 1$ を n で割った商である。ゆえに

$$(7) \quad U_\ell = U_{i,j}, \quad j = \lfloor (\ell - 1)/n \rfloor + 1, \quad i = \text{mod}(\ell - 1, n) + 1.$$

余談 B.1 例えば C 言語のプログラムならば、 ℓ から i, j を求めるために

```
j=(ell-1)/n; i=(ell-1)%n+1;
```

とすれば良い。もっとも、割り算をする必要はあまりない。 ℓ をループの制御変数にするのではなく、 i と j をループの制御変数にすればよいから。つまり

```
for (ell=1; ell<=(m*n); ell++) {
    i=(ell-1)%n+1;
    j=(ell-1)/n+1;
    U[ell] = i と j の式;
}
```

と書く代わりに

```
for (i=1; i<Nx; i++) { // i<Nx は i<=m と書く方が分かりやすい?
    for (j=1; j<Ny; j++) { // j<Ny は j<=n と書く方が分かりやすい?
        ell=i+n*(j-1);
        U[ell]= i と j の式;
    }
}
```

とすれば良い。■

Laplacian $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ には、2階微分が現れるが、それを2階中心差分近似すると、 $-\Delta u = f$ は

$$-\left[\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h_x^2} + \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h_y^2} \right] = f(x_i, y_j)$$

という差分方程式で置き換えられる。

$$(8) \quad AU = \mathbf{f},$$

$$(9) \quad A = I_m \otimes \left[\frac{1}{h_x^2} (2I_m - J_m) \right] + \left[\frac{1}{h_y^2} (2I_n - J_n) \right] \otimes I_m,$$

$$(10) \quad \mathbf{f} = (f_\ell), \quad f_\ell = f(x_i, y_j).$$

```
% W, H, Nx, Ny に値が適当に記憶されているとして
% 例: W=2; H=1; Nx=8; Ny=4;
hx=W/Nx;
hy=H/Ny;
m=Nx-1;
n=Ny-1;
ex=ones(m,1);
ax=spdiags([-ex 2*ex -ex],-1:1,m,m)/(hx*hx);
ey=ones(n,1);
ay=spdiags([-ey 2*ey -ey],-1:1,n,n)/(hy*hy);
Im=speye(m,m);
In=speye(n,n);
a=kron(ay,Im)+kron(In,ax);
```

なお、 U_{ij} を並べる際に、(6) でなく

$$(11) \quad U_\ell = U_{i,j}, \quad \ell = m(i-1) + j$$

とする場合もある。むしろこちらの方が多いかもしいない。MATLAB で `meshgrid()` などを使うには、こちらの方が都合が良い(本文の例を参照せよ)。この場合は

$$(12) \quad j = \text{mod}(\ell - 1, m) + 1, \quad i = \lfloor (\ell - 1)/m \rfloor + 1.$$

また連立1次方程式の係数行列は

$$(13) \quad A = I_m \otimes \left[\frac{1}{h_y^2} (2I_n - J_n) \right] + \left[\frac{1}{h_x^2} (2I_m - J_m) \right] \otimes I_n.$$

コードの方は最後の `a=kron(ay,Im)+kron(In,ax);` を

```
a=kron(Im,ay)+kron(ax,In);
```

に置き換えれば良い。

B.3 コピペして試す

(i, j) 成分を指定して行列を作る、という例を一つくらい。

Hilbert 行列 myhilbert.m

```
% うっかり作ってしまったけれど、hilb(n) という関数が標準で用意されている。
function a=myhilbert(n)
    a=zeros(n,n);
    for i=1:n
        for j=1:n
            a(i,j)=1/(i+j-1);
        end
    end
end
```

5 次の Hilbert 行列の逆行列と条件数を求めてみる。

```
edit myhilbert
```

(上のコードを入力して保存した後)

```
h=myhilbert(10)
inv(h)
cond(h)
```

(逆行列の成分はほぼ 10^{12} 程度の大きさになり、条件数もほぼ 1.6×10^{13} 程度になる。— 非常に大きい、つまり悪条件。)

1次元ラプラシアン、逆行列や LU 分解を求めてみる。

1次元ラプラシアン (Dirichlet 境界条件)

```
L=1;
N=5;
h=L/N;
e=ones(N-1,1);
a=spdiags([-e 2*e -e],-1:1,N-1,N-1)/(h*h);
full(a)

b=inv(a);
full(b)

[l u]=lu(a);
full(l)
full(u)
```

2次元ラプラシアン (Dirichlet 境界条件)

```
W=2;
H=1;
Nx=8;
Ny=4;
hx=W/Nx;
hy=H/Ny;
m=Nx-1;
n=Ny-1;
ex=ones(m,1);
ax=spdiags([-ex 2*ex -ex],-1:1,m,m)/(hx*hx);
ey=ones(n,1);
ay=spdiags([-ey 2*ey -ey],-1:1,n,n)/(hy*hy);
Im=speye(m,m);
In=speye(n,n);
a=kron(ay,Im)+kron(In,ax);
full(a)
```

ans =

```
64 -16 0 0 0 0 0 -16 0 0 0 0 0 0 0 0 0 0 0 0 0
-16 64 -16 0 0 0 0 0 -16 0 0 0 0 0 0 0 0 0 0 0 0
0 -16 64 -16 0 0 0 0 -16 0 0 0 0 0 0 0 0 0 0 0 0
0 0 -16 64 -16 0 0 0 0 0 -16 0 0 0 0 0 0 0 0 0 0
0 0 0 -16 64 -16 0 0 0 0 0 -16 0 0 0 0 0 0 0 0 0
0 0 0 0 -16 64 -16 0 0 0 0 0 -16 0 0 0 0 0 0 0 0
-16 0 0 0 0 0 0 64 -16 0 0 0 0 0 -16 0 0 0 0 0 0
0 -16 0 0 0 0 0 -16 64 -16 0 0 0 0 0 -16 0 0 0 0 0
0 0 -16 0 0 0 0 0 -16 64 -16 0 0 0 0 0 -16 0 0 0 0
0 0 0 -16 0 0 0 0 -16 64 -16 0 0 0 0 0 -16 0 0 0 0
0 0 0 0 -16 0 0 0 0 -16 64 -16 0 0 0 0 0 -16 0 0 0
0 0 0 0 0 -16 0 0 0 0 -16 64 -16 0 0 0 0 0 -16 0 0
0 0 0 0 0 0 -16 0 0 0 0 -16 64 -16 0 0 0 0 0 -16 0
0 0 0 0 0 0 0 -16 0 0 0 0 -16 64 -16 0 0 0 0 0 -16
0 0 0 0 0 0 0 0 -16 0 0 0 0 -16 64 -16 0 0 0 0 0
0 0 0 0 0 0 0 0 0 -16 0 0 0 0 -16 64 -16 0 0 0 0
0 0 0 0 0 0 0 0 0 0 -16 0 0 0 0 -16 64 -16 0 0 0
0 0 0 0 0 0 0 0 0 0 0 -16 0 0 0 0 -16 64 -16 0 0
0 0 0 0 0 0 0 0 0 0 0 0 -16 0 0 0 0 -16 64 -16 0
0 0 0 0 0 0 0 0 0 0 0 0 0 -16 0 0 0 0 -16 64 -16
0 0 0 0 0 0 0 0 0 0 0 0 0 0 -16 0 0 0 0 -16 64
```

>>

C INTLAB

C.1 いんとろ

C.2 入手とインストール

Paypal で支払うのだけど、事務を通すのが面倒だ。

zip ファイルを展開して、出て来たフォルダーを ~/ Documents/MATLAB の下に移動する。
それから

最初に一度だけこれを実行

```
>> cd Intlab_V9
>> startintlab
```

(何回か Enter する。)

~/Documents/MATLAB/startup.m

```
addpath('~/Documents/MATLAB/Intlab_V9');
```

C.3 簡単な使い方

そういえば、過去に説明を書いたことがない。

Rump 氏によると、Rump [2] は

「INTLAB の原論文です。INTLAB を用いた結果を論文にするときには必ずこの文献を参照しましょう。」

とのこと。

Rump の学生 Hargreaves 氏の修士論文 [3] に、INTLAB のチュートリアルが含まれているとか (いかにもドイツですね)。

Moore-Kearfott-Cloud [4] も参考になるかも。

私の周囲の人が作ったプログラム例としては、福澤誠人君の修士論文 [5] がある。

D row major vs. column major

配列はメモリーの中で連続した領域を使って記憶されている。

1次元配列 a の場合は、アドレスの低い (数値として小さい) 方から、 $a(1), a(2), \dots$ が並んでいる。これは大抵のプログラミング言語に共通である。

2次元配列の場合にメモリーの中でどう並ぶかは、2つの流儀がある。

D.1 MATLAB, Fortran は column major order

MATLAB の場合、例えば $a=\text{zeros}(m,n)$; で用意した a について、アドレスの低い方から

$$\begin{aligned} &a(1,1), a(2,1), \dots, a(m,1), \\ &a(1,2), a(2,2), \dots, a(m,2), \\ &\quad \quad \quad \dots, \\ &a(1,n), a(2,n), \dots, a(m,n) \end{aligned}$$

と、行 (row) 番号が先に動くように並んでいる。これを “column major order” という。

MATLAB 互換の Octave, Scilab 以外に、Fortran や R, S-Plus, Julia など column major order である。

D.2 C は row major order

一方、C 言語で `double a[m][n]`; で定義した `a` について、アドレスの低い方から

$$\begin{aligned} & a[0][0], a[0][1], \dots, a[0][n-1], \\ & a[1][0], a[1][1], \dots, a[1][n-1], \\ & \dots, \\ & a[m-1][0], a[m-1][1], \dots, a[m-1][n-1] \end{aligned}$$

と、列 (column) 番号が先に動くように並んでいる。これを “row major order” という。C 以外に、C++, Mathematica, Pascal などでも row major order である。

FORTTRAN は column major order, C は row major order

- 2次元の差分法での説明で、筆者は $U_{i,j}$ を $u(1,1), u(2,1), \dots$ と並べたが、これは column major order ということになる。
- MATLAB で

```
% a=0; b=3; c=0; d=1; nx=30; ny=10;
x=linspace(a,b,nx+1);
y=linspace(c,d,ny+1);
[X,Y]=meshgrid(x,y);
size(X)
size(Y)
X(:)
Y(:)
```

とすると、`X` と `Y` のサイズはともに、 $(ny + 1, nx + 1)$ である。 $X(j, i)$ と $Y(j, i)$ ($1 \leq i \leq nx + 1, 1 \leq j \leq ny + 1$) は、それぞれ格子点 $P_{i-1, j-1} = (x_{i-1}, y_{j-1})$ の x 座標 x_{i-1} 、 y 座標 y_{j-1} を記憶している、

`X` についていうと、メモリー中で

$$\begin{aligned} X(1,1) &= x_0, X(2,1) = x_0, \dots, X(ny+1,1) = x_0, \\ X(1,2) &= x_1, X(2,2) = x_1, \dots, X(ny+1,2) = x_1, \\ & \dots, \\ X(1,nx+1) &= x_{nx}, X(2,nx+1) = x_{nx}, \dots, X(ny+1,nx+1) = x_{nx} \end{aligned}$$

と並んでいる。

`Y` についていうと、メモリー中で

$$\begin{aligned} Y(1,1) &= y_0, Y(2,1) = y_1, \dots, Y(ny+1,1) = y_{ny}, \\ Y(1,2) &= y_0, Y(2,2) = y_1, \dots, Y(ny+1,2) = y_{ny}, \\ & \dots, \\ Y(1,nx+1) &= y_0, Y(2,nx+1) = y_1, \dots, Y(ny+1,nx+1) = y_{ny} \end{aligned}$$

と並んでいる。

`meshc(X,Y,U)` のようにして描画する場合、 U は X, Y と同じような順番でデータを格納する必要がある。これはつまり

$$U_{0,0}, U_{0,1}, U_{0,2}, \dots, U_{0,ny}, U_{1,0}, U_{1,1}, U_{1,2}, \dots, U_{1,ny}, \dots, U_{nx,0}, U_{nx,1}, U_{nx,2}, \dots, U_{nx,ny}$$

という順に並べるということである。

$i \in \{0, 1, \dots, nx\}, j \in \{0, 1, \dots, ny\}$ とするとき

$$\ell = i + j(ny + 1)$$

で定まる ℓ は $\{0, 1, (nx + 1)(ny + 1) - 1\}$

E グラフィックス

- `close(番号)` で特定のウィンドウを閉じる。`close all` で全てのウィンドウを閉じる。
- `fig=gcf; figure(fig)` あるいは、`figure(gcf)` でウィンドウが `visible` になる。
- `title(文字列)` でウィンドウのタイトルを描く。

```
title(['heat equation, t=', num2str(tau*n, '%4.2f')])
```

- `axis([x1 x2 y1 y2])`

F 固有値関係の命令

MATLAB のオンライン・マニュアルの日本語版には、長いこと直されていない誤訳がある。“magnitude” を無視しているけれど、これは「絶対値」という意味である。

参考文献

- [1] 桂田祐史：Poisson 方程式に対する差分法, <http://nalab.mind.meiji.ac.jp/~mk/labo/text/poisson.pdf> (2000 年?～).
- [2] Rump, S.: INTLAB - INTerval LABoratory, in Csendes, T. ed., *Developments in Reliable Computing*, pp. 77–104, Kluwer Academic Publishers, Dordrecht (1999), <http://www.ti3.tuhh.de/rump/>.
- [3] Hargreaves, G. I.: Interval analysis in MATLAB, Master’s thesis, Manchester Institute for Mathematical Sciences, School of Mathematics, The University of Manchester (2002), <http://www.ti3.tuhh.de/rump/intlab/narep416.pdf>.
- [4] Moore, R. E., Kearfott, R. B. and Cloud, M. J.: *Introduction to INTERVAL ANALYSIS*, SIAM (2009), <http://www-sbras.nsc.ru/interval/Library/InteBooks/IntroIntervAn.pdf> から入手できる。
- [5] 福澤誠人：定常 Stokes 方程式の有限要素解の事後誤差評価と事前誤差評価, <http://nalab.mind.meiji.ac.jp/~mk/labo/report/open/2004-fukuzawa.pdf> (2005).