MATLABの使い方

小林 哲則
koba@waseda.jp

小川 哲司*
ogawa.tetsuji@waseda.jp

*本年度の問い合わせ窓口は小川です．必要に応じてメールにて連絡してください．

行列の入力

行列を入力するには次のように書く：

```
>> S = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

MATLABは，入力された行列を次のように表示する：

```
>> S

 S =

     16    3    2    13

     5    10   11    8

     9    6    7    12

     4    15   14    1
```

行列の要素

行列の i, j 要素を指定するためには次のように書く：

S(i, j)

例えば、先の行列で **S(4,2)** は **15** である.

よって,

>> S(1,4) + S(2,4) + S(3,4) + S(4,4)

と書けば,

ans =

34

となる.

注意

定義されていない行列要素を参照するとエラーとなる.

先の例では，４×４の要素しか定義されていない（値が与えられていない）ので次のような指定はエラーとなる.

```
>> temp = S(5,5)
```

一方，定義されていない行列要素でも，値を代入する場合には，行列が拡張されて，その要素を使えるようになる.

```
>> temp =  S;

>> temp(4,5) = 71

temp  =
     16    3    2    13    0
     5    10    11    8    0
     9    6    7    12    0
     4    15    14    1    71
```

（4×5のマトリクスとなる）

変数

MATLABでは，型や次元数の宣言は一切不要である．

新たな変数名を見つけると，自動的に変数を作り領域を割り当てる．

既に変数が存在するときは値を変更し，必要であれば領域を割り当て直す．

例えば，

```
>> num_integer = 100
```

は， **num_integer** という名前の1×1行列 を作り，値 **100** を入れる．

変数名は，英字で始まる任意の英数字列（下線を含む）で構成される．

MATLAB は大文字と小文字を区別する．

変数に割り当てられた行列を見るためには，単に変数名を書く．

```
>> num_integer
 num_integer  =
              100
```

数

通常の10進数表記を採用している.

3　　-199　　0.00002　　1.2345667

10のべき乗を表すために，**e** を用いる.

1.31e-31　　1.563e24

虚数を表すために，**i**，**j** を用いる.

0 + 1i　　3 - 2i　　1.25e5i

特別な定数

i	虚数単位 **sqrt(-1)**
pi	円周率 **P = 3.1415927 .....**
eps	マシンの精度
nan	非数値
inf	無限大（最大値）

コロンオペレータ(１)

コロン **:** は，様々な形で使われる.

例えば,

```
>> 1:10
```

と書けば，**1**から始まり**10**で終わる行ベクトルを表す.

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

増分を1以外にするためには，増分パラメータを指定する.

例えば,

```
>> 10:-2:0                % 初期値：増分：最終値
```

と書けば，増分は **-2** と解釈される。

```
ans =
     10   8   6   4   2   0
```

コロンオペレータ(２)

行列の中で

S(i : j, k)

と書けば，k番目の列ベクトルのi番目からj番目までの列ベクトルを表す.

```
>> S(2:3,1)

ans  =
     5
     9
```

同様に,

```
>> S(1, 2:4)
```

と書けば,

```
ans  =
     3    2    13
```

を得る.

コロンオペレータ（３）

行列の要素の位置で，：を単独で用いると，行あるいは列ベクトルを表す．

S(：,k)

と書けば，k 番目の列ベクトルを表す．

```
>> S(:,1)

ans  =
      16
       5
       9
       4
```

同様に，

```
>> S(1,:)
```

と書けば，第1行ベクトルを得る．

```
ans  =
      16    3    2   13
```

出力の制御

単に命令を書けば，結果を出力する．

```
>> s = rand(1, 5, 'normal')
 s  =
      0.5608486   0.6623569   0.7263507   0.1985144   0.5442573
```

セミコロンで終われば，結果は出力されない．

```
>> s = rand(1, 5, 'normal')；
```

継続行

1行を超える場合は，改行の前に３つのピリオド **...** を入れ，命令を次の行に続ける.

例えば,

```
>> s = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 ...
>> - 1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

結合

小さな行列をまとめて大きな行列を作ることができる.

例えば,

```
>> a = [1 2 3];

>> b = [4 5 6];

>> c = [7 8 9];

>> d = [a b c]

 d  =

     1   2   3   4   5   6   7   8   9
```

行／列の削除(１)

```
>> s =
     1   2    3    4
     5   6    7    8
     9   10   11   12
```

に対し，　２列目の列ベクトルを削除するには，

```
>> s(:,2) = [ ]
```

とする．

```
 s =
     1   3    4
     5   7    8
     9   11   12
```

行列に対し，単一の要素は指定できない．例えば，以下はエラーとなる．

```
>> s(1,3) = [ ]
```

行／列の削除(２)

しかし，添え字をひとつにすると，ひとつの要素を削除し， 残りの要素を列ベクトルに変換する.

例えば，行列 **s** を列ベクトルにした上で，第2要素を削除する場合は以下のように書く.

```
>> s(2) = [ ]

s  =
     1
     9
     2
     6
     10
     3
     7
     11
     4
     8
     12
```

行／列の削除(３)

行列 **s** を列ベクトルにした上で，第2要素から第9 要素まで，2つおきに削除する場合は以下のように書く．

```
>> s(2:2:9) = [ ]

 s  =

      1
      9
      6
      3
     11
      4
      8
     12
```

行列を作る関数(１)

要素が全て零である行列（零行列）を作るには **zeros(N, M)** を使う.

```
>> zeros(3, 2)

ans  =
     0   0
     0   0
     0   0

>> T  = [1 2 3 4; 5 6 7 8]

T  =
     1   2   3   4
     5   6   7   8

>> zeros(T)

ans  =
     0   0   0   0
     0   0   0   0
```

行列を作る関数(２)

要素が全て1である行列を作るには **ones(N, M)** を使う.

```
>> ones(2, 3)

ans  =
     1   1   1
     1   1   1
```

一様乱数を要素とする行列を作るには **rand(N, M, 'uniform')** を使う.

正規乱数を要素とする行列を作るには **rand(N, M, 'normal')** を使う.

```
>> rand(2, 4, 'normal')
ans  =
    - 1.2858605    0.6107784  - 0.3778182    2.5749104
     0.5971163  - 1.0567863    1.1456173  - 0.5005553
```

セル配列

セル配列は多次元配列で，その要素は異なる配列から構成される．

空行列のセル配列は，**cell()** 関数で作成する．

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> M = cell(1, 4);
>> M{1, 1} = A;
>> M{1, 2} = sum(A, 1);
>> M{1, 3} = sum(A, 2);
>> M{1, 4} = sprintf('matrix');
>> M

M  =
    {3x3 double}    {1x3 double}    {3x1 double}    {'matrix'}

>> M{1, 2}

ans  =
    12    15    18
```

演算子

MATLABの行列演算

[]	行列の定義，結合	;	行の区切り
()	要素の参照 m=a(k)	()	要素への代入 a(k)=m
’	転置	+	加算
-	減算	*	乗算
¥	左からの除算	/	右からの除算
^	べき乗	.*	要素毎の掛け算
.¥	要素毎の左除算	./	要素毎の右除算
.^	要素毎のべき乗	.*.	クロネッカの乗算
.¥.	クロネッカの左除算	./.	クロネッカの右除算

演算例

演算子 * は行列の掛け算を表す.

```
>> A

A =
     1   2
     3   4

>> B

B =
     2   1
     1   3

>> A * B

ans =
     4    7
    10   15
```

演算例

演算子 **.*** は**要素ごとの**掛け算を表す.

```
>> A

A =
     1   2
     3   4

>> B

B =
     2   1
     1   3


>> A .* B

ans =
     2   2
     3   12
```

演算例

演算子 **.*.** は**テンソル積（クロネッカー積）**を表す.

```
>> A

A =
     1   2
     3   4

>> A .*. A

ans =
     1   2    2    4
     3   4    6    8
     3   6    4    8
     9   12   12   16
```

$$A.*.B = \begin{bmatrix} A(1,1)*B & \cdots & A(1,n)*B \\ \vdots & \ddots & \vdots \\ A(m,1)*B & \cdots & A(m,\ n)*B \end{bmatrix}$$

演算例

列ベクトルを使って，関数表を作ることができる．

```
>> s = [1:6]';


>> [s  log10(s)]

 ans  =
        1.0000         0
        2.0000    0.3010
        3.0000    0.4771
        4.0000    0.6021
        5.0000    0.6990
        6.0000    0.7782
```

スカラーと行列の演算

行列とスカラーが混在する場合，様々な規則で演算が生じる．

例えば，行列からスカラーを引く場合，各々の要素が引かれる．

```
>> s = ones(4,4)

 s  =

      1   1   1   1
      1   1   1   1
      1   1   1   1
      1   1   1   1

>> s-1

 ans  =

      0   0   0   0
      0   0   0   0
      0   0   0   0
      0   0   0   0
```

関数：sum関数（１）

```
>> S = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]

S =
     16    3    2   13
      5   10   11    8
      9    6    7   12
      4   15   14    1
```

なる行列 **S** に対して，関数 **sum(S,1)** は，行列要素を**列ごとに**足し合わせ，**行ベクトル**として返す．

```
>> sum(S, 1)

ans =
     34   34   34   34
```

関数：sum関数（２）

```
>> S = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]

S =
    16    3    2    13
     5   10   11     8
     9    6    7    12
     4   15   14     1
```

なる行列 **S** に対して，関数 **sum(S,2)** は，行列要素を**行ごとに**足し合わせ，**列ベクトル**として返す.

```
>> sum(S, 2)

ans =
    34
    34
    34
    34
```

関数：mean関数（１）

```
>> S = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]

S =
     16    3    2   13
      5   10   11    8
      9    6    7   12
      4   15   14    1
```

なる行列 **S** に対して，**mean(S,1)** 関数は，行列要素に対して**列ごとに**平均値を算出し，**行ベクトル**として返す．

```
>> mean(S, 1)

ans =
    8.5   8.5   8.5   8.5
```

関数：mean関数（２）

```
>> S = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]

S =
    16    3    2    13
     5   10   11     8
     9    6    7    12
     4   15   14     1
```

なる行列 **S** に対して，**mean(S,2)** 関数は，行列要素に対して**行ごとに**平均値を算出し，**列ベクトル**として返す．

```
>> mean(S, 2)

ans =
    8.5
    8.5
    8.5
    8.5
```

関数：find 関数（１）

find 関数は，配列の列ベクトルを順に並べて作ったベクトルの中の，条件に合う**要素のインデックスの集合**を，**行ベクトル**として返す．

```
>> a = [1 -2 3; 4 5 -6; 7 8 -9]

a  =
     1  -2   3
     4   5  -6
     7   8  -9

>> find(a<0)

ans  =
     4   8   9

>> a(find(a<0)) = 0

ans  =
     1   0   3
     4   5   0
     7   8   0
```

関数：find関数（２）

```
>> W

W =
   1   2   3   4   5   6   7   8   9   0
   2   3   4   5   6   7   8   9   0   1
   0   0   0   0   0   1   1   1   2   2
```

なる行列 W において，3行目が1である1，2行目の値を行列 X に代入する

.

```
>> idx = find(W(3, :) == 1)

idx =
     6   7   8.

>> X = W(1:2, idx)

X =
     6   7   8.
     7   8   9.
```

（　X = W(1:2, find(W(3,:) == 1))　）

関数：repmat

repmat(M,m) により，行列**M**は行列 **ones(m,m) .*. M** に置き換えられる.

repmat(M,m,n) により，行列**M**は行列 **ones(m,n) .*. M** に置き換えられる.

```
>> W

  W  =
     1   2   3   4   5   6   7   8   9   0
     2   3   4   5   6   7   8   9   0   1
     0   0   0   0   0   1   1   1   2   2

>> repmat(W(2,:), 2, 1)

 ans =
     2   3   4   5   6   7   8   9   0   1
     2   3   4   5   6   7   8   9   0   1
```

その他の関数

size(a) ：行列 **a** の次元数を表す行ベクトルを返す.

length(a)：行列 **a** の要素数を返す.

sin(a) ：各々の要素のsin

fft(a) ：FFT

ifft(a) ：逆FFT

abs(a) ：複素数の大きさ

inv(a) ：逆行列

det(a) ：行列式

条件分岐：if 文

```
if  condition
        % code
elseif  condition
        % code
else
        % code
end
```

【例】

```
if X > 0
        disp('X is positive.');
elseif X < 0
        disp('X is negative.');
else
        disp('X is zero.');
end
```

条件分岐：select 文

```
select  condition
        case 1
                % code
        case N
                % code
end
```

【例】

```
select  modulo(num,2)
        case 0
                disp('The Number is Even');
        case 1
                disp('The Number is Odd');
        case 2
                disp('Inavlid Number');
        end
```

繰り返し：while 文

```
while condition
        % code
        % loop counter i.e., count =count +1;
end
```

【例】

```
mat = rand(1, 10, 'normal');
binary = zeros(mat);
count = 1;
while (count <= length(mat))
        if mat(count) >= 0
                binary(count) = 1;
        else
                binary(count) = -1;
        end
        count = count + 1;
end
```

繰り返し：break 文

【例】

```
mat = rand(1, 10, 'normal');
binary = zeros(mat);
count = 1;
while(count <=  length(mat))
        if mat(count) >= 0
                binary(count) = 1;
        else
                binary(count) = -1;
        end
        count = count + 1;

        if count == 5    % break condition
                break;
        end
end
```

繰り返し：for 文

```
for    loop condition
       % code
end
```

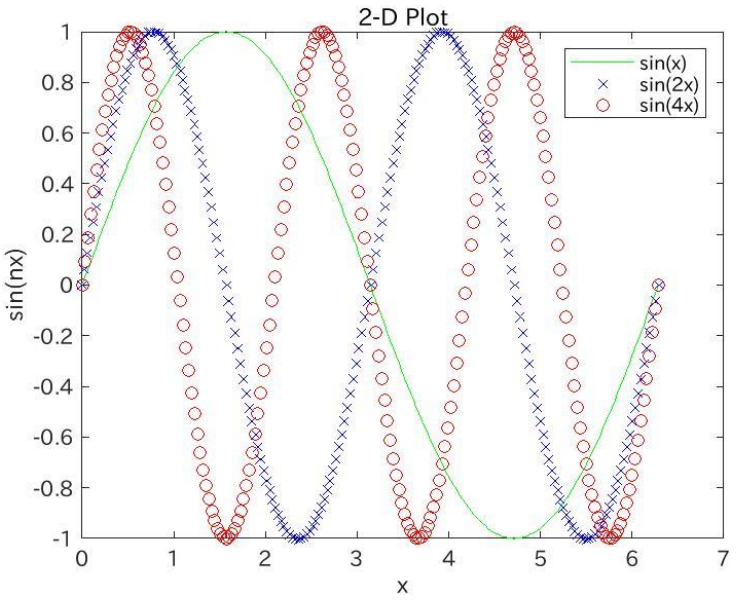
【例】

```
mat = rand(1, 10, 'normal');
binary = zeros(size(mat));
for count = 1: length(mat)
       if mat(count) >= 0
              binary(count) = 1;
       else
              binary(count) = -1;
       end
end
```

2次元プロット

```
X = (0:pi/100:2*pi)’;
Y1 = sin(X);
Y2 = sin(2 * X);
Y3 = sin(4 * X);

figure;
plot(X, Y1, 'g', X, Y2, 'bx', X, Y3, 'ro');
title('2-D Plot');                          % 図タイトル
xlabel('x');                                % x軸ラベル
ylabel('sin(nx)');                          % y軸ラベル
legend('sin(x)', 'sin(2x)', 'sin(3x)');     % 凡例
```



ファイル入出力

以前にMATLABで作った変数をMATファイル（.mat）に保存したり，保存されたMATファイルを読み込んで変数を再利用できる.

テキストデータは，次のような矩形型の表形式でなければならない.

```
1 2 3
4 5 6
```

【例】 変数 A をMATファイル example.mat に保存する.

```
>> A = [16.0 3.0; 2.0 13.0];
>> save('example', 'A');
```

【例】MATファイル example.mat に保存された変数 A を読み込む.

```
>> clear A;
>> load('example', 'A');
```

テキスト・ファイルへの書き込み

数値のテーブルを temp.txt という名前のテキストファイルに書き込む.

```
X = 1:5;
Y = X.^2;
Z = [ X; Y ];
fid = fopen('temp.txt', 'w');
fprintf(fid, '%2s%5s¥n', 'x', 'x.^2');
fprintf(fid, '%2d%5d¥n', Z);
fclose(fid)；
```

temp.txt には以下の内容が書き込まれる.

```
>> type temp.txt

x x.^2
1    1
2    4
3    9
4   16
5   25
```

テキスト・ファイルの読み込み

A = fscanf(fid, format, sizeA)

は，識別子 **fid** のファイルから **format** で指定した形式に従い，次元が **sizeA** の配列 **A** にファイルデータを読み込む.

```
>> fid = fopen('temp.txt', 'r');

>> A = fscanf(fid, '%d%d', [2 inf]);

>> fclose(fid);

>> A'

ans =
     1    1
     2    4
     3    9
     4   16
     5   25
```

オーディオファイルを読み書きする

```
[y, Fs] = audioread('filename');
```

は，**filename** というファイルからデータを読み取り，サンプルデータ **y** と，そのデータのサンプリング周波数 **Fs** を返す.

```
>> [y, Fs] = audioread('sample.wav');      % WAVEファイルを読み込み
>> sound(y,Fs);                           % オーディオを再生する
```

```
audiowrite(y, Fs, 'filename');
```

は，オーディオデータ **y** の行列を，サンプリング周波数 **Fs** で **filename** というファイルに書き込む.

スクリプトファイル

コマンドを保存したテキストファイル（**.m**）を実行できる.

1. **スクリプトの作成：**新しいスクリプトは以下の方法で作成できる.

- **[ホーム]**タブにある**[新規スクリプト]** ボタンをクリックし，以下の命令を記述する．その後，**scriptsine.m** という名前で保存する.

```
% script m-file to plot sine wave
time = 0 : 0.01 : 20;
plot(sin(time));
```

- 関数 **edit** を使用してエディタを起動し，上記の様にコードを記述する.

```
>> edit scriptsine.m
```

2. **コードの実行：**以下の方法を使用してコードを実行できる.

- コマンドラインからスクリプト名を入力する.

```
>> scriptsine
```

- **[エディター]**タブにある**[実行]**ボタンをクリックする.

関数を作る

関数の作り方

```
function 出力 = 関数名(入力)
        % code
end
```

【例】**定義例**：関数と同じ名前の独自ファイル（**test.m**）に保存する.

```
function [out1, out2] = test (in)
        out1 = -in;
        out2 = in * in;
end
```

【例】**使用例**：コマンドラインから関数を呼び出すことが可能になる.

```
>> [a,b] = test(3)

 a
        ans = -3
 b
        ans = 9
```