

Matlabの使い方

作成者: 岩本祐太郎
(陳延偉)

MATLAB

■ MATLAB:

行列ベースの高水準科学技術計算用プログラミング言語

■ MATLABの特徴:

- インタプリタ形式(一行ずつプログラムを実行できる)
- 豊富なライブラリ(toolbox)のサポート
- 自然なベクトル・行列演算
- 充実したオンラインhelp (関数毎にページがある)
 コマンドウィンドウで『help 関数名』からも調べられる

■ MATLABが利用されている分野

機械学習、信号処理、画像処理、通信、金融制御など

MATLABの導入方法

立命館大学 RAINBOW ITサポート

MATLABソフトウェア配布

下記URL:

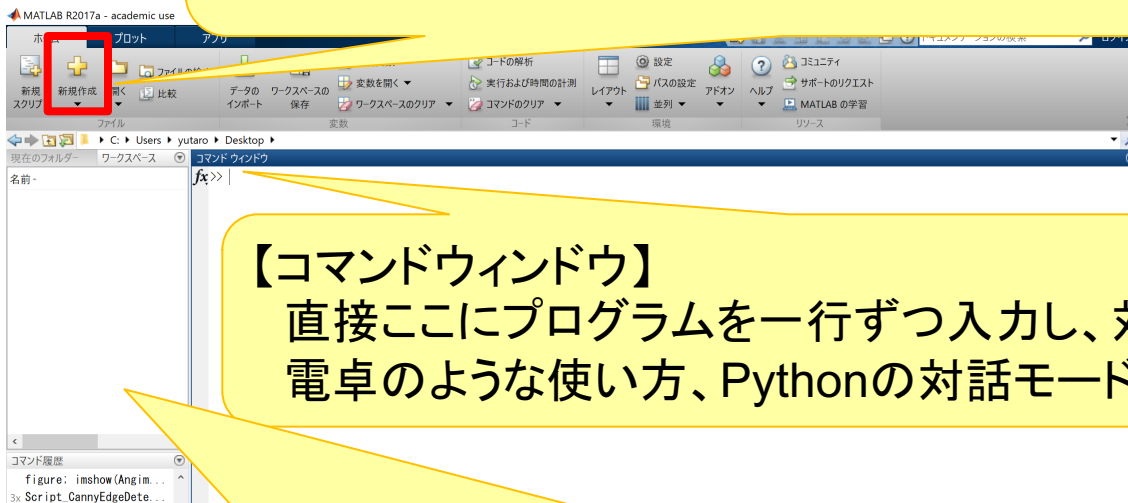
<http://www.ritsumeit.ac.jp/rainbow/service-softwarematlab/>

学生用「マニュアル」を読みMATLABをインストール

MATLABの起動

【スクリプトファイル(*.m)の作成】

プログラムコードの記述、保存する。課題ではスクリプトファイルにコードを記述する。.c, .py, .jsファイルと同じ。



【コマンドウィンドウ】

直接ここにプログラムを一行ずつ入力し、対話的に実行できる。電卓のような使い方、Pythonの対話モードに近い。

【ワークスペース】

現在まで実行中のプログラム変数が保存されている。

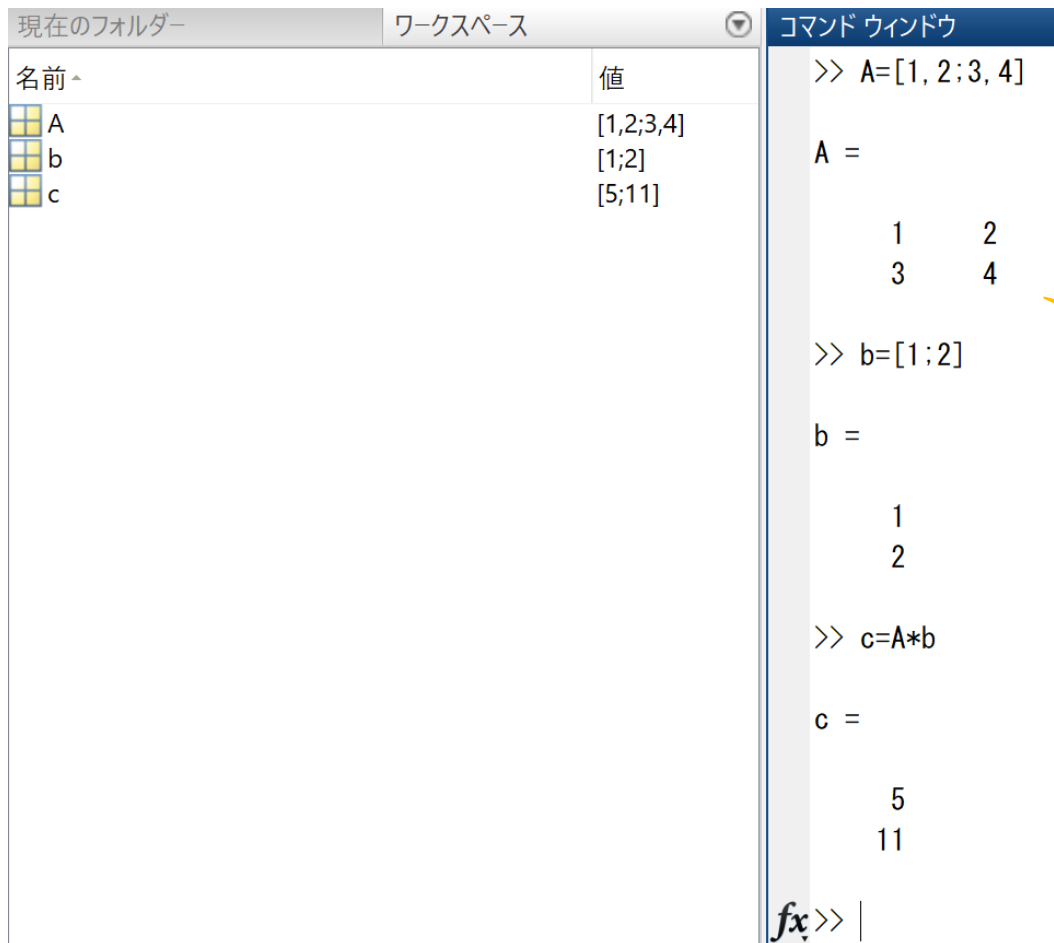
MATLABの利用

- コマンドライン上での利用

例:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\mathbf{c} = \mathbf{A}\mathbf{b} = \begin{bmatrix} 5 \\ 11 \end{bmatrix}$$



The screenshot shows the MATLAB interface. The 'ワークスペース' (Workspace) window on the left lists variables A, b, and c with their values. The 'コマンド ウィンドウ' (Command Window) on the right shows the execution of MATLAB commands: A=[1, 2;3, 4], b=[1;2], and c=A*b. The output for A is shown as a 2x2 matrix, and the output for c is shown as a 2x1 column vector. The Command Window prompt is 'fx>> |'.

名前	値
A	[1,2;3,4]
b	[1;2]
c	[5;11]

```
>> A=[1, 2;3, 4]

A =

     1     2
     3     4

>> b=[1;2]

b =

     1
     2

>> c=A*b

c =

     5
    11









fx>> |
```

ベクトル、行列の記述
カンマ(,): 次の列に
セミコロン(;): 次の行に
アポストロフィー('): 転置

変数のデータ型

- double: 倍精度配列(8バイト(64bit), デフォルト)
- uint8: 8bit符号なし整数配列(画像読み込み(imread)のデフォルト)
- char: 文字配列

ワークスペース内の値をダブルクリックすると、変数の中身を確認できる。**A**は型がデフォルトのdoubleになっていることがわかる。画像変数imgは256x256のuint8であることが確認できる。

現在のフォルダー		ワークスペース	
名前 ^		値	
	A		[1,2;3,4]
	b		[1;2]
	c		[5;11]
	img		256x256 uint8

A		2x2 double	
		1	2
1		1	2
2		3	4

配列へのアクセス

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

- $A(\text{行番号}, \text{列番号})$ で取得する。⇒ **インデックスが1から始まる**ことに注意
 $A(1,1) = 1, A(1,2) = 2, A(2,1)=3, A(2,2)=4$
- $A(\text{番号}) \Rightarrow$ **列毎に順番に読み込む**ことに注意
 $A(1) = 1, A(2) = 3, A(3) = 2, A(4) = 4$

条件分岐

コマンド ウィンドウ

```
>> a=3
```

```
a =
```

```
3
```

```
>> if a>2
```

```
disp(a)|
```

```
end
```

```
3
```

```
fx>> |
```

if 条件文
実行内容

end

※ dispは変数の表示

特殊な演算子

- . * 要素毎の乗算
- . / 要素毎の右除算
- ^ 行列のべき乗
- . ^ 要素毎のべき乗
- ' 転置

繰り返し処理

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
>> for i=1:2  
for j=1:2  
disp(A(i, j))  
end  
end
```

for (例:i=1:2)

処理内容

end

1

2

3

4

1回目のループでi=1,2回目のループでi=2の二
回の繰り返しで処理が繰り返される

関数の利用

- 様々な関数を利用することができる

関数の利用方法

- inv: 逆行列の計算
- eig: 固有値と固有ベクトルの計算
- svd: 特異値分解の計算
- tan: ラジアン単位の引数の正接
- tand: 度単位の引数の正接
- atand: 度単位の逆正接

画像処理に関する関数

`imread('パス')` パスの画像を読み込み

`imwrite(uint8(A),'パス')` 画像Aの保存

`figure` 図用ウィンドウの作成

`imshow(A, [])` 画像Aの表示

`plot(x,y)` 線形2次元プロット(x:配列, y:配列, 各座標点)

下記関数など直接レポート課題を解く関数はレポートでは使用禁止

`fspecial(type)` type型の2Dフィルタの作成(ガウシアン,sobelなど)

`imfilter(A,b)` 画像Aにbをフィルタリング(rewitt, sobelなど)

`edge(A,'type')` 画像Aにtype型のエッジ検出(cannyなど)

スクリプトファイルと関数の作成

同一フォルダ内で別の関数ファイルを作成

Demo.m

```
a = 3;  
b = 4;  
  
c = func1(a,b);  
  
Disp(c);
```

func1.m

```
function output = func1(in1, in2)  
  
output = 10*in1+in2;
```

出力

```
>> Demo  
34
```

サンプルコード(画像反転)



原画像



結果画像

■ inverselImage.mファイル

%画像の読み込み

```
img = double(imread('lena.bmp')); % .mファイルと同じフォルダに画像を入れること
```

%反転画像を生成する方法

%方法1 matlab

```
invImg1 = 255-img;
```

%自動的に255-各画素を計算

%方法2 C言語に近い方法

```
[H,W] = size(img);
```

%画像サイズの取得

```
invImg2 = zeros(H,W);
```

%出力画像領域の確保

```
for i=1:H
```

%for文: iが1からHまで回る

```
    for j = 1:W
```

%for文: jが1からWまで回る

```
        invImg2(i,j) = 255-img(i,j); %matlabでは配列は1から始まる
```

```
    end
```

```
end
```

```
figure; imshow(invImg1, []); %画像の表示
```

%一つのウィンドウに複数の画像を表示

```
figure;
```

```
subplot(1,2,1); imshow(img, []); title('オリジナル画像'); %subplot(行,列,何番目)
```

```
subplot(1,2,2); imshow(invImg2, []); title('結果画像');
```

```
imwrite(uint8(invImg1), 'inverseImg.bmp'); %画像の保存, ※uint8形式にすること
```

サンプルコード(画像を明るくする)



原画像



結果画像(1.5倍明るく)

■ brightImage.mファイル

%画像の読み込み

```
img = double(imread('lena.bmp')); % .mファイルと同じフォルダに画像を入れること
```

%画像を明るくする方法

%方法1 matlab

```
rate = 1.5;
```

```
brightImg = funcbright(img, rate); %funcbright関数で明るくする
```

```
figure; imshow(brightImg, []); %画像の表示
```

```
imwrite(uint8(brightImg), 'brightImg.bmp'); %画像の保存
```

■ funcbright.mファイル

```
function output = funcbright(img, rate)
```

%function 出力変数名 = 関数名(入力関数名)

%出力関数が複数ある場合は[a, b, c]のようにすること

```
output = rate.*img; % .*はimgの各要素にrateの値を掛けることを意味する
```

```
output(output>255) = 255; %outputの中身が255を超える値は255にする
```

MATLABの便利な機能

F5: プログラムを実行(ブレークポイントまで)

F10: 一行ずつ実行

ワークスペース: 現在のプログラム実行位置での変数と中身を確認できる。
変数をクリックするとその変数の中身を見ることができる。

ブレークポイント: 「-」マークをクリックすると設定される

The screenshot displays the MATLAB IDE interface. On the left, the 'Workspace' window shows variables: H (512), img (512x512 uint8), invImg1 (512x512 uint8), invImg2 (512x512 uint8), and W (512). On the right, the 'Code Editor' shows a script with several lines of MATLAB code. Annotations in yellow callouts point to specific features: one points to the 'Workspace' window, another points to a red circle (breakpoint) on line 3, and a third points to a green arrow (current execution position) on line 13. The code includes comments in Japanese and English, such as '%画像の読み込み' and '%画像サイズの取得'.

名前	値
H	512
img	512x512 uint8
invImg1	512x512 uint8
invImg2	512x512 uint8
W	512

```
1  invImg1 = zeros(H, W);  
2  %画像の読み込み  
3  img = imread('lena.bmp'); % .mファイルと同じフォルダに画像を入れること  
4  
5  
6  
7  invImg1 = zeros(H, W); %自動的に255 - 各画素を計算  
8  
9  %方法: 逆の方法  
10 [H, W] = size(img); %画像サイズの取得  
11 invImg2 = zeros(H, W); %出力画像領域の確保  
12 invImg2 = uint8(invImg2); %double型からuint8型に変更  
13 for i = 1:H %for文: iが1からHまで回る  
14     for j = 1:W %for文: jが1からWまで回る  
15         invImg2(i, j) = 255 - img(i, j); %matlabでは配列は1から始まる  
16     end
```