

MATLABの基本的な使い方

MATLABについて

- MATLABとは、
 - 科学技術計算のための高性能プログラミング言語
- 特徴
 - ・ 配列が基本データ型
 - ・ ベクトル(1次元配列)、行列(2次元配列)
 - ・ 対話的システム
 - ・ 豊富な関数ライブラリとグラフィックツール
 - ・ 同様の機能を持ったフリーソフトのScilab、Octaveがある
- 使用される主な分野
 - 数値計算、シミュレーション、信号処理、グラフィックス等

MATLABの基本的な使い方1

- ◆ コマンドウィンドウにプログラムを打ち込み、リターン。

```
>> a=1, b=2, c=a+b ↵
```

```
a =
```

```
1.
```

```
b =
```

```
2.
```

```
c =
```

```
3.
```

- ◆ 行末に3つのピリオドを入力すると改行可能

```
>> a=1+2+3 ... ↵
```

```
+4+5 ↵
```

```
a =
```

```
15
```

MATLABの基本的な使い方2

◆ M-fileにプログラムを保存して実行 (コンパイルやリンクは必要ない)

```
>> edit filename ↵
```

とコマンドウィンドウに入力すると、エディタが立ち上がる

例) >> edit test1.m ↵

test1.m (エディタで下記を入力後、保存)

```
function c = test1(a,b)
c = a + b
```

コマンドウィンドウで関数を実行する

```
>> test1(a,b) ↵
```

```
ans =
```

```
3.
```

MATLAB 記述ルール

◆ コメントなど

- % コメントを表す(%以降が行末までコメントになる)。

- ; 結果の非表示

式の終わりにセミコロン「;」をつける

例) >> a=1, b=3; ↵

- 大文字と小文字

MATLABでは、C言語と同様に大文字と小文字を区別する

ベクトル・行列の生成方法

- 列の区切り: 半角スペース(またはカンマ)
- 行の区切り: ; 「セミコロン」

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

```
x =
```

```
1.  
2.  
3.
```

```
>> A=[1 2 3; 4 5 6; 7 8 10]
```

```
A =
```

```
1.  2.  3.  
4.  5.  6.  
7.  8. 10.
```

- 関数`linspace`を使用して、初期値、最終値、要素数を指定し、ベクトルを生成

```
>> linspace(1,9,5)
```

```
ans =
```

```
1.  3.  5.  7.  9.
```

配列要素の指定

- Aのi行j列の要素はA(i,j)と指定

A =

```
1.  2.  3.  
4.  5.  6.  
7.  8. 10.
```

```
>> A(2,2)
```

ans =

```
5.
```

- A(m,:)でm行の要素、A(:,n)でn列の要素を指定

```
>> A(2,:)
```

ans =

```
4.  5.  6.
```

```
>> A(:,1)
```

ans =

```
1.
```

```
4.
```

```
7.
```

行列の演算

■行列A, Bについて、行列演算ができる。

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

```
A =
```

```
1. 2.
3. 4.
```

```
>> B=[5 6 ;7 8]
```

```
B =
```

```
5. 6.
7. 8.
```

■足し算 +

```
>> A + B
```

```
ans =
6. 8.
10. 12.
```

■引き算 -

```
>> A - B
```

```
ans =
-4. -4.
-4. -4.
```

■掛け算 *

```
>> A * B
```

```
ans =
19. 22.
43. 50.
```

注2) Macの場合、バックスラッシュは option+¥

注) ¥マークは、Windows以外では
バックスラッシュ\で表示される

■左除算 ¥

(inv(A) * Bとほぼ同じ)

```
>> A ¥ B
```

```
ans =
-3. -4.
4. 5.
```

■右除算 /

(A * inv(B)とほぼ同じ)

```
>> A / B
```

```
ans =
3. -2.
2. -1.
```

■Aのべき乗 ^

```
>> A^2
```

```
ans =
7. 10.
15. 22.
```

■共役転置 '

```
>> A'
```

```
ans =
1. 3.
2. 4.
```

演算子の前にピリオドを付けると
要素ごとの演算になるものが
多い。A.*Bなど試してみよう

比較演算子・論理演算子

- 条件式を作成する際に用いる。
オペランド(演算対象)が行列の場合、演算は要素ごとに適用され、結果も行列となる。

種類	記号	意味
比較演算子	>	より大きい
	<	より小さい
	>=	以上
	<=	以下
	==	等しい
	~=	等しくない(最初の記号はチルダ)
論理演算子	&&	かつ
		または (shift + ¥キーを2回入力)

関数とは？

- MATLABでは、色々な関数ができる。
 - 組み込み関数(元から用意されている関数)
 - ◆ $\sin(x)$, $\log(x)$ など
 - ユーザー関数(自分で作成する関数)
 - インライン関数

■ 関数の使い方

```
>> x = 1.0; y = sin(x)
```

```
>> [y1,y2,...] = func(x1,x2,...)
```

xの値が**入力**されて、
yに結果が**出力**される。
つまり、 $\sin(1.0)$ の値がyに
代入される。

ユーザー関数の作成方法

1. エディタを開く。 `>> edit test2`
2. 関数の内容を自分で書く(次ページで説明)。

```
function [x,y] = test2(a,b,c)
x = a + b;
y = a - c;
```

注) コマンドウィンドウで使った変数をそのまま関数の中で使うことはできません。(同じ名前の変数にしても、別々のものになる)

3. 保存する(ファイル名は、**test2.m**のようにする)。
4. コマンドウィンドウで関数を使ってみる。

```
>> a = 5; b = 2; c = 1; [r,s] = test2(a,b,c)
```

補足) ユーザー関数の中から、組み込み関数や他のユーザー関数を呼び出すこともできます。

ユーザー関数の内容

`function`は
関数の作成時に
必要な宣言

関数名: `test2`
(ファイル名と一致させる)

2行目以降に
内容を書く。

```
function [x,y] = test2(a,b,c)
x = a + b;
y = a - c;
```

出力する変数(出力引数と言う)も
好きなだけ増やせる。
ただし、関数の中で値が定まる
必要がある。

入力する変数(入力引数と言う)は
好きなだけ増やせる。

注) 引数は「ひきすう」と読みます。

練習問題：簡単な関数を作成する

- ◆ aとbを入力したら、 $a + b$, $a - b$, $a \times b$, a / b の結果をすべて出力する関数test3を作成してみよう。

初等関数の使い方

■MATLABでは、多数の組み込み関数を使用できる。ベクトルや行列の入力にも対応するものが多い。

```
>> x=[1 2]
```

```
x =
```

```
1. 2.
```

■正弦

```
>> sin(x)
```

```
ans =
```

```
0.8414710 0.9092974
```

■余弦

```
>> cos(x)
```

```
ans =
```

```
0.5403023 -0.4161468
```

■正接

```
>> tan(x)
```

```
ans =
```

```
1.5574077 -2.1850399
```

■対数関数

```
>> log(x)
```

```
ans =
```

```
0. 0.6931472
```

■指数関数

```
>> exp(x)
```

```
ans =
```

```
2.7182818 7.3890561
```

行列演算に関する関数の使い方

■ 様々な行列演算

```
A =  
 1.  2.  
 3.  4.
```

■ 逆行列

```
>> R=inv(A)  
R =  
 -2.  1.  
 1.5 -0.5
```

■ m行n列で要素がすべて1の行列

```
>> d=ones(2,3)  
d =  
 1.  1.  1.  
 1.  1.  1.
```

■ m行n列で要素がすべて0の行列

```
>> d=zeros(2,1)  
d =  
 0.  
 0.
```

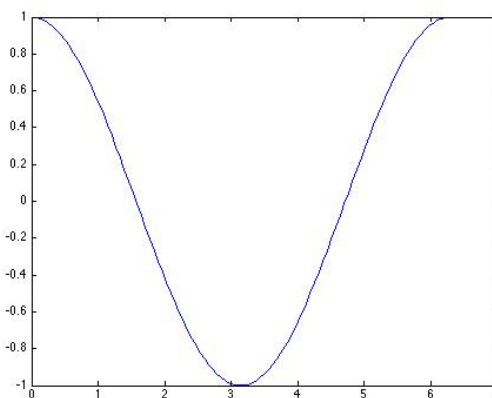
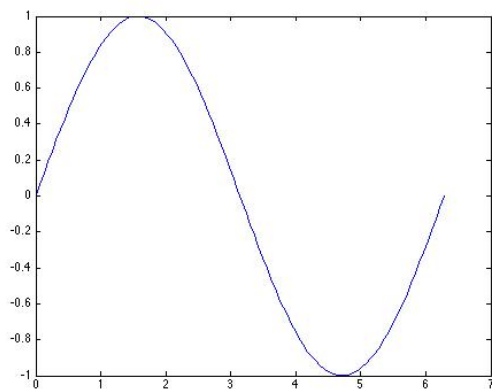
■ m行n列の単位行列

```
>> E=eye(2)  
E =  
 1.  0.  
 0.  1.
```

2次元グラフィックス(1)

- plotで、2次元のグラフを表示。グラフはFigure1というウィンドウに表示される。
※plotの度に上書きされる。

```
>> x=linspace(0,2*pi,100);  
>> plot(x,sin(x));  
>> plot(x,cos(x));
```

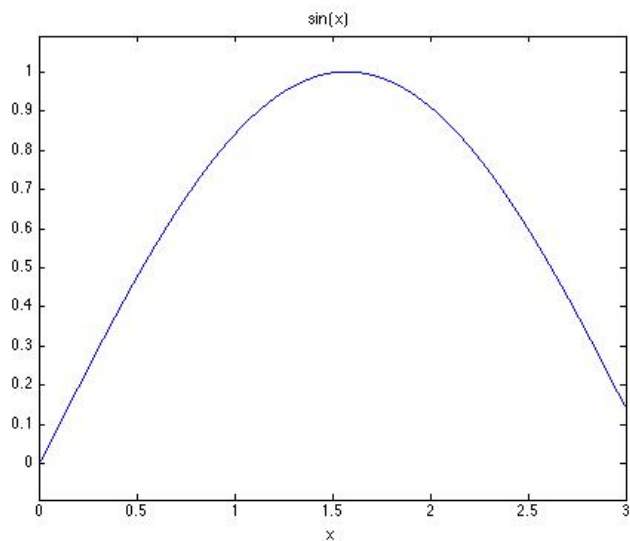


- Figure1を「Save as」で保存(別名で保存)。
jpg形式などで保存することが可能。

2次元グラフィックス(2)

- ezplotで、関数のグラフを表示。グラフはFigure1というウィンドウに表示される。
※hold on で重ねてグラフを表示可能。(hold offで解除)

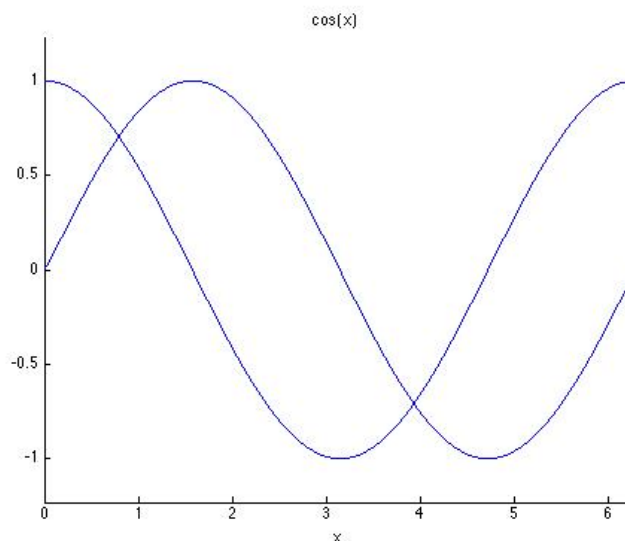
```
>> ezplot('sin(x)',[0,3])
```



```
>> hold on
```

```
>> ezplot('sin(x)',[0,6.28])
```

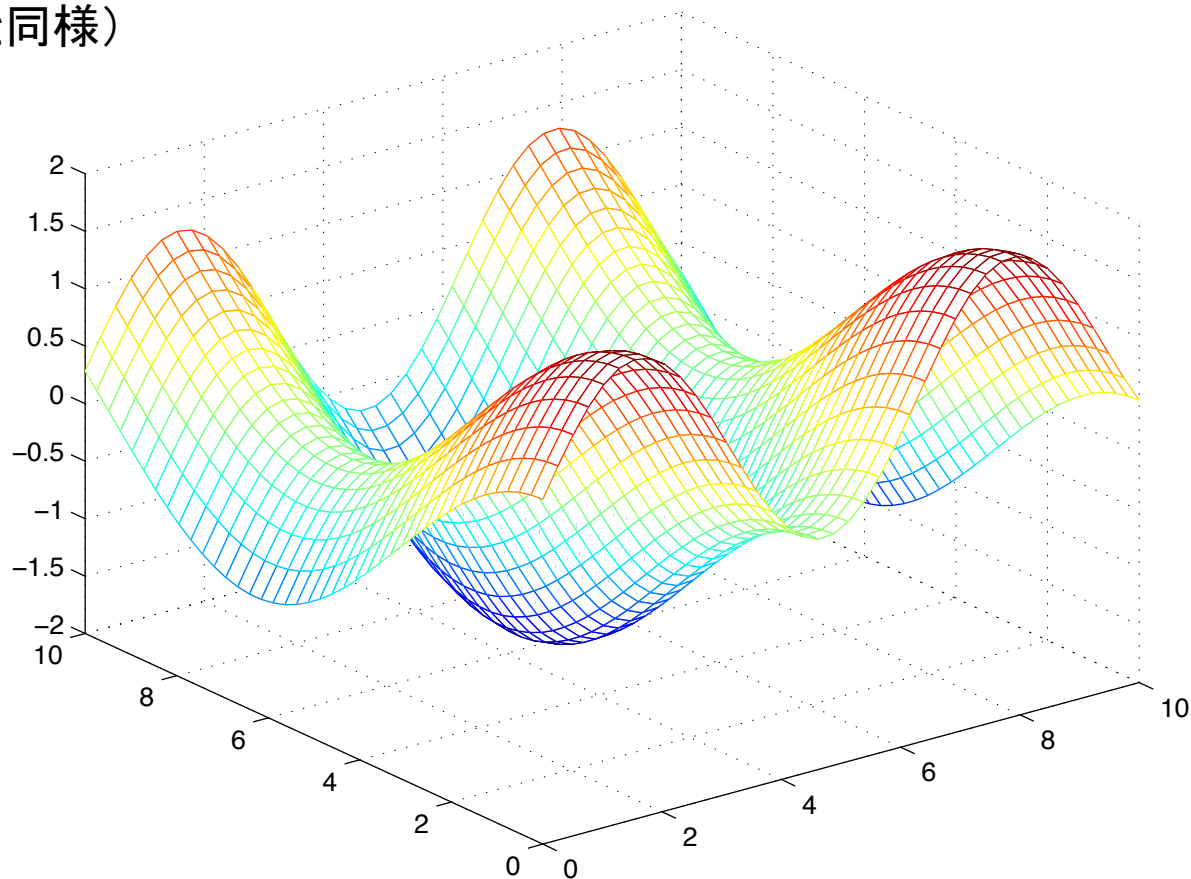
```
>> ezplot('cos(x)',[0,6.28])
```



3次元グラフィックス(1)

■tの範囲を、「meshgrid」関数でxy座標に変換。
描画したい関数「 $z=\sin(x)+\cos(y/2)$ 」を計算し、
「mesh」関数で3次元グラフィックスを表示
(グラフの保存方法は、2次元と同様)

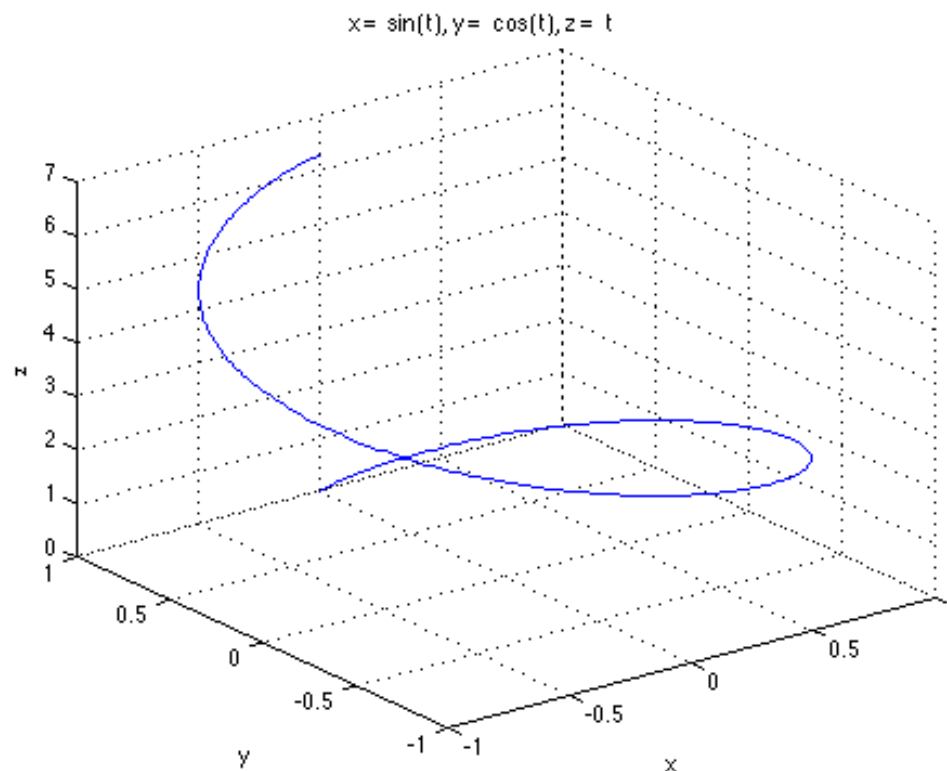
```
>> t=linspace(0,10,40);  
>> [x,y]=meshgrid(t,t);  
>> z=sin(x)+cos(y/2);  
>> mesh(x,y,z);
```



3次元グラフィックス(2)

■「ezplot3」関数で簡単な3次元グラフィックスを描画可能。
ezplot3(x,y,z,[min,max])は、領域 $\min < t < \max$ で、
曲線 $x = x(t)$, $y = y(t)$, $z = z(t)$ をプロットします。
(デフォルトの領域 $0 < t < 2\pi$)

```
>> ezplot3('sin(t)','cos(t)','t')
```



【参考】関数一覧: 行列生成関数

関数名	説明
zeros	要素がすべてゼロの行列を作成
ones	要素がすべて1の行列を作成
eye	単位行列を作成
diag	入力がベクトルの場合は対角行列を作成 入力が行列の場合は、対角要素を抽出
magic	1 から n^2 までの整数を使用して、行方向と列方向の和が等しくなる n 行 n 列の行列を作成
rand	$A = \text{rand}(n)$ は、开区間 $(0,1)$ 上の標準一様分布から導き出される疑似乱数値を含む n 行 n 列の行列を作成
randn	$A = \text{randn}(n)$ は、標準正規分布から導き出される疑似乱数値を含む n 行 n 列の行列を作成
linspace	$y = \text{linspace}(a,b)$ は、 a と b の間で、線形に等間隔な 100 点の行ベクトル y を作成 $y = \text{linspace}(a,b,n)$ は、 a と b の間で、線形に等間隔な n 点の行ベクトル y を作成

【参考】関数一覧: 行列関数

関数名	説明
\backslash (mldivide)	$X = A \backslash B$ で、行列方程式 $AX = B$ を X について解く。 $X = \text{inv}(A) * B$ とほぼ同じ。 (\backslash マークは、Windows以外ではバックスラッシュ\で表示される)
$/$ (mrdivide)	$X = A / B$ で、行列方程式 $XB = A$ を X について解く。 $X = A * \text{inv}(B)$ とほぼ同じ。
det	行列式を計算
rank	行列のランクの計算
lu	$[L,U,P] = \text{lu}(A)$ は、 $P*A = L*U$ となる下三角行列 L 、上三角行列 U 、置換行列 P を計算
qr	$[Q,R] = \text{qr}(A)$ は、 A が m 行 n 列とすると、 $A = Q*R$ となるような、 m 行 n 列の上三角行列 R と m 行 m 列の直交行列(複素の場合はユニタリ行列) Q を計算
eig	$d = \text{eig}(A)$ は、行列 A のすべての固有値を計算。 $[X,D] = \text{eig}(A)$ は、固有ベクトルを並べた行列 X と固有値を並べた対角行列 D を計算。

【参考】関数一覧: 初等関数など

関数名	説明
sin	正弦
exp	指数関数(自然対数の底 e のべき乗)
log10	対数(底が10) ※logは自然対数
sqrt	平方根
round	要素を最も近い整数に丸める
min	最小値 ・Aがベクトルの場合、min(A) は A の最小要素を返す ・A が行列の場合、min(A) は、A の列をベクトルとして取り扱い、各列の最小要素を含む行ベクトルを返す
max	最大値 ・Aがベクトルの場合、max(A) は A の最大要素を返す ・A が行列の場合、max(A) は、A の列をベクトルとして取り扱い、各列の最大要素を含む行ベクトルを返す。
mean	平均値 ・Aがベクトルの場合、mean(A) は A の平均値を返す ・A が行列の場合、mean(A) は、A の列をベクトルとして取り扱い、平均値からなる行ベクトルを返す
std	標準偏差 ・s = std(X) は X がベクトルのとき、上記の (1) を使用して標準偏差を計算 ・Xが行列の場合、std(X) は X の各列の要素の標準偏差を含む行ベクトルを返す
sum	総和 ・Aがベクトルの場合、sum(A) は要素の和を計算 ・A が行列の場合、sum(A) は A の列をベクトルとして扱い、各列の和を行ベクトルとして返す

プログラミング

■MATLABでプログラミングを行う方法を説明する。
前述した組み込み関数や演算も組み合わせ、
プログラム(Mファイル)を簡単に作成することができる。

- ・スクリプトによるプログラミング
MATLABの命令を並べておくだけでプログラムを作る方法。
実行するときは、コマンドウィンドウに、ファイル名を入力する。
- ・関数を定義することによるプログラミング
新しい関数を定義することによって、組み込み関数と同様に
使用することができる。

注) 関数名とファイル名は一致させる必要がある。
たとえば、関数名がfunc1なら、ファイル名はfunc1.mのようにする。

プログラミング例

例1) 連立一次方程式 $Ax=b$ の解を求めるスクリプト

```
>> edit linear1.m
```

```
A=[3 2 7; 1 6 3; 1 0 6];  
b=[3 ;-2 ;1];  
x=A \ b
```

Macの場合は、option+¥

コマンドウィンドウで実行

```
>> linear1
```

```
x =  
 1.6000  
-0.5500  
-0.1000
```

例2) ベクトルxの平均値を求める関数

```
>> edit mean1.m
```

```
function y = mean1(x)  
n = length(x);  
y = sum(x)/n;
```

コマンドウィンドウで実行

```
>> x=[1 4 7 10]
```

```
x =  
 1 4 7 10  
>> mean1(x)  
ans =  
 5.5000
```


制御（繰り返し、条件分岐）

■プログラムをする際に重要となる制御文

制御文	機能	使用例
for	指定回数の繰り返し	for ループ変数=ループの範囲 命令 end
while	不定回数の繰り返し	while 終了条件 命令 end
if, elseif, else	条件実行	if 条件式 命令 end
switch, case	条件式に基づき、case文で実行を切り替え	switch 変数 case 条件 命令 otherwise 命令 end

for文

■ for文は繰り返す回数・範囲等繰り返しの設定をします。

■ 典型的な形式(1) XをYからZまで1ずつ増加させながら処理を繰り返す

```
for X=Y:Z  
    (処理)  
end
```

(例) iを1から10まで1ずつ増加させながらiの表示を繰り返す。
for i=1:10
 disp(i); % disp(x)は変数xを画面に表示する関数
end

■ 典型的な形式(2) XをYからZまでWずつ増加させながら処理を繰り返す

```
for X=Y:W:Z  
    (処理)  
end
```

(例)

- for i=1:2:10 (1から10まで2刻みで増加させ、10を超えない間は繰り返す)
- for i=10:-1:1 (10から1まで-1刻みで減少させ、1になるまで繰り返す)

for文

- 同じような操作を繰り返すときに使う。

for1.m

```
for i=1:n
    disp(i);
end
```

```
>> n = 10;
>> for1
```

for2.m

```
for i=n:-1:1
    disp(i);
end
```

```
>> n = 10;
>> for2
```

- ベクトルの総和の計算

sum1.m

```
function s = sum1(a)
n = length(a);
s = 0;
for i=1:n
    s = s + a(i);
end
```



```
>> x = [1 4 5 2 7];
>> t = sum1(x)
```

for文の練習問題

1. nを設定したとき、コマンドウィンドウに1からnまでを1つおきに表示するスクリプト for3.m を作成しよう。

```
>> n=7;  
>> for3  
1  
3  
5  
7
```

2. 1のスクリプトをもとに、nを入力とする関数 for4 を作成しよう。
3. ベクトルの内積を計算する関数 dot1 を作成して、正しく動くか試してみよう。

while文

■while文は、**継続条件**を指定し、不定回数の繰り返しを行います。

(例) eps1.m (1.0 からつぎに大きい浮動小数点数までの距離を確かめるスクリプト)

```
Eps = 1;  
  
while 1+Eps > 1  
    Eps = Eps/2;  
end  
  
Eps = Eps*2
```

Epsが1より大きい間は、
Eps/2を処理し続ける

実行結果

```
>> eps1
```

```
Eps =
```

```
2.2204e-16
```

if文

■条件式は、変数と演算子の組み合わせとして表現されます。(演算子はp.9参照)

(例)もし変数 x が10以上の時、 k の値を1増加させる

```
if x >= 10
    k = k + 1;
end
```

■if文をさらに細かく制御したい場合、else や elseif を使用する

もし条件1が真であれば、命令1を実行、
条件1が偽であれば、条件2を見て、真であれば、命令2を実行、
さらに条件2も偽であれば、命令3を実行

```
if 条件1
    命令1
elseif 条件2
    命令2
else
    命令3
end
```

if文

↓ こちらはエディタで
打ち込む

↓ こちらはコマンドウィンドウで
打ち込む

if1.m (スクリプトの例)

```
if a > 4
    disp(b);
end
```



```
>> a = 3; b = 5;
>> if1
```

if2.m (関数の例)

```
function c = if2(a,b)
if a > b
    c = a + b;
else
    c = a - b;
end
```



```
>> a = 3; b = 5;
>> c = if2(a,b)
```

switch文

■式に基づき case 間で実行を切り替えます

(例) evenodd.m 偶数と奇数を確認する関数

```
function evenodd(n)

switch mod(n,2)
  case 0
    disp('even number');
  case 1
    disp('odd number');
  otherwise
    disp('error');
end
```

除算の剰余

mod(x,y) は、x mod yを出力します

2で割った剰余が0の時、even(偶数)
剰余が1の時、odd(奇数)

それ以外はエラーとする

インライン関数(1)

■MATLABではインライン関数を自分で定義できる。

たとえば、 $f(x) = e^x \sin(10x)$ を定義する場合は

```
>> f=inline('exp(-x)*sin(10*x)')
```

のようにすれば良い。

このとき、 $f(3)$ は

```
>> f(3)  
ans = -0.0492
```

のように計算できる。

インライン関数(2)

■インライン関数のベクトル化

前ページの $f(x)$ では、入力 x がベクトルのときにエラーになってしまう。
そこで、 $f(x)$ の定義を下記のように少しだけ変更する(*の前にピリオドを付ける)。

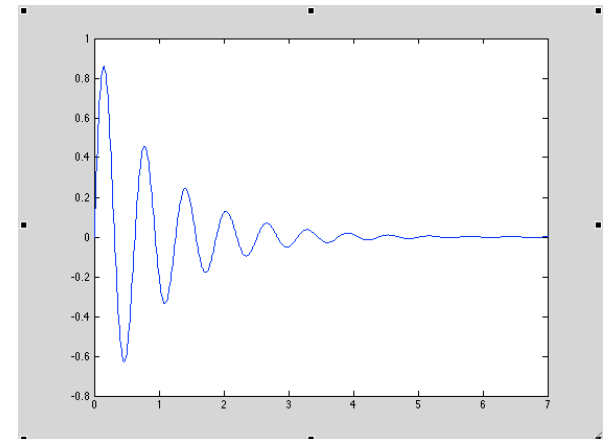
```
>> f=inline('exp(-x).*sin(10*x)')
```

このようにすると、入力 x がベクトルであっても対応できる。

```
>> x=[1 2 3];  
>> f(x)  
ans = -0.2001    0.1236   -0.0492
```

最後に、 $f(x)$ のグラフ(定義域は $[0,7]$ とする)を描いてみよう。

```
>> x=0:0.01:7;  
>> plot(x,f(x))
```



参考文献

1. 大石 進一: MATLABによる数値計算, 培風館.
2. 櫻井 鉄也: MATLAB/Scilabで理解する数値計算, 東京大学出版会.