TR - H - 142

MATLAB 入門

玉川 浩

1995. 4. 17

ATR人間情報通信研究所

〒619-02 京都府相楽郡精華町光台2-2 ☎ 0774-95-1011

ATR Human Information Processing Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-774-95-1011

Facsimile: +81-774-95-1008

目次

1 はじめに

本編は、データの取り込み－加工－視覚化という一連のデータ処理の作業を想定し、MATLAB を最小限の労力で習得するための入門書です。

MATLAB は MATrix LABoratry という名前のとおり、行列演算が手軽にできるツールですが、数値処理、強力なグラフィック機能、簡単なGUI も備えたツールで、一般的なデータ処理から結果の視覚化までを簡単なコマンドで実現できます。

使用するデータファイルは、アスキー(ascii)形式のテキストで、以下のように各フィールドがスペースもしくはタブで区切られ、各レコードは、改行で区切られたファイルを対象とします。

右記の例では、65 5657 -3573 が1つのレコードになります。またこのレコードの第2フィールドは 5657 です。対象となるデータが、複数のファイルにまたがっていたり、フィールドの数が異なる場合は、あらかじめ、UNIX のコマンド(head、tail、paste、nawk 等)で、フィールドの数をそろえ、1つのファイルに結合してください[1]。

```
% cat wavedata.ascii
65   5657    -3573
47   5611    -2994
57   5547    -2414
21   5472    -1878
     ...
```

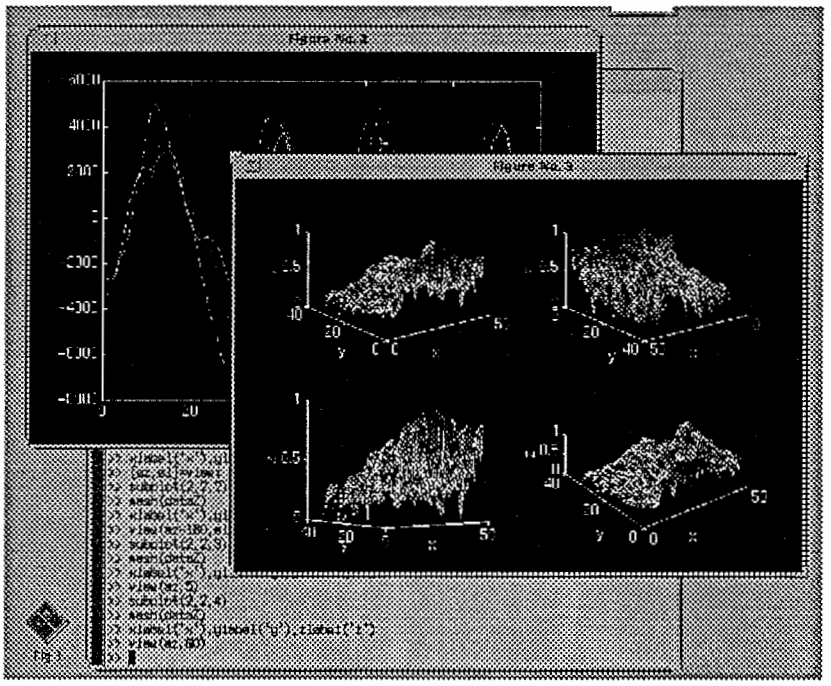


図1: MATLAB のメイン画面 (kterm 上) とグラフィック画面

*UNIX は UNIX Systems Laboratories,Inc. の登録商標です。
*Matlab は The MathWorks,Inc. の登録商標です。
*PostScript は Adobe Systems Incorporated の登録商標です。

[1] head は、指定された数だけファイルの先頭からレコードを取り出すコマンド、paste は、指定された複数のファイルのフィールドを結合するコマンドです。

2 起動と終了

2.1 環境設定

MATLAB を起動する前に、環境変数やパスを設定しておきます。

環境変数 $MATLAB には MATLAB のホームディレクトリを入れます。

また、MATLAB 自身の実行ファイルがあるディレクトリも $path に入れておきましょう。

```
% setenv MATLAB /usr/hearing/matlab
% set     path=($path $MATLAB/bin)
```

2.2 起動

MATLAB を起動するには、以下のコマンドを入力します。

MATLAB が起動されて、入力を促すプロンプト (>>) が表示されます。MATLAB は、この入力行から入力されたコマンドを解釈し実行します。

```
% matlab

                    < M A T L A B (tm) >
    (c) Copyright 1984-92 The MathWorks, Inc.
                    All Rights Reserved
                        Version 4.2
                        Mar 29 1994

Commands to get started: intro, demo, help help
Commands for more information: help, whatsnew, info,
subscribe


>> ■
```

2.3 終了

MATLAB を終了するには、MATLAB の入力行に quit を入力します。

セッションの間に使用された CPU の使用時間が表示されて、シェルのプロンプトに戻ります。

```
>> quit

   328309 flops.


% ■
```

以後の章では、MATLAB が起動されている状態で説明を行います。また、説明の中で行列とあった場合、特に断わりのない限り、列ベクトルや行ベクトルも含みます。

2.4 ヘルプ

演算	MATLAB での表現	備考
ヘルプ	help *command-name*	
コマンド／関数の検索	lookfor *string*	

コマンドの使い方や、コマンド名が判らなくなったり、どんなコマンドがあるのか知りたい場合、MATLAB には、ヘルプや検索のコマンドが用意されています。

□ ヘルプ

help のあとにコマンド名を入力すると、コマンドに関する説明が表示されます。

```
>> help quit

 QUIT   Terminate MATLAB.
        QUIT terminates MATLAB.
```

help だけ入力すると、MATLAB の項目一覧と各項目に関する簡単な説明が表示されます。

```
>> help

.                    -  General purpose commands.
matlab/general       -  General purpose commands.
matlab/ops           -  Operators and special characters.
                ...
toolbox/optim        -  Optimization Toolbox.
toolbox/signal       -  Signal Processing Toolbox.

For more help on directory/topic, type ''help topic''.
```

help のあとに項目名を指定すると、その項目に関するコマンドの一覧が表示されます。

```
>> help general

 General purpose commands.

 Managing commands and functions.
   help    - On-line documentation.
   what    - Directory listing of M-, MAT- and MEX-files.
   type    - List M-file.
             ...
```

□ コマンドの検索

コマンドを検索するには、lookfor というコマンドを使います。

FFT に関するコマンドを検索します。

```
>> lookfor fft
FFT      Discrete Fourier transform.
FFT2     Two-dimensional Fast Fourier Transform.
FFTSHIFT Move zeroth lag to center of spectrum.
IFFT     Inverse discrete Fourier transform.
            ...
```

[1] ヘルプの説明の中ではコマンドや引数は大文字で書かれてありますが、入力するときは、特別な場合を除いて小文字で入力してください。MATLAB は大小文字を区別します。

3　データ処理の基礎

3.1　数値演算

演算	MATLAB での表現	備考
演算子	+ - * / ^	
式の区切り	, ; 改行	
コマンドの実行	*command-name* arg1 arg2 ...	*ex.* help demo
関数の呼出し	*func-name*(arg1,arg2, ...)	*ex.* max(3,4)

□ 式の入力と結果

MATLAB の入力行は、^F (1 文字先へ)、^B (1 文字後へ)、^P (前のコマンド)、^N (次のコマンド)、^A (行の先頭へ)、^E (行の最後へ)、^K (削除) などのコントロールコードで編集ができますので、emacs や tcsh を使っている人には扱いやすいでしょう。

式を入力して改行を行うと、ans = の後に結果が表示されます。

```
>> 3+2
ans =
     5
```

式をカンマ (,) で区切って一行に並べることもできます。

```
>> 3+2,3*(5+6)
ans =
     5

ans =
    33
```

結果を表示したくない場合、式をセミコロン (;) で区切ると、結果が表示されません。

```
>> 3+2;
>>
```

□ 変数

a に値を代入し、a を用いて計算します。

```
>> a=6
ans =
    6

>> a+7
ans =
    13
```

□ コマンドの実行と関数の呼出し

コマンドの実行　*command-name* arg1 arg2 ...
関数の呼出し　[ret1,ret2,...]=*function-name*(arg1,arg2,...)

```
>> help sqrt

 SQRT   Square root.
        ...

>> y=sin(1);
```

2 つ以上の値を返す関数の例です。
size は行列の行数、列数を調べる関数です。
詳しくは行列の操作を参照してください。

```
>> [m,n]=size(1)
m =
     1

n =
     1
```

□ 定数

MATLAB で用意されている定数として、虚数単位(i または j)の他に、円周率(pi)、無限大(inf)、Not-a-Number(NaN)、直前の結果を示す ans などがあります。

```
>> pi
ans =
    3.1416

>> 1000^1000
ans =
   Inf
```

□ 複素数

複素数を表すには、i またはj を使用します。

```
>> 3+2*i
ans =
   3.0000 + 2.0000i
```

ただし、定数は変数として使用することもできます。

```
>> i=2;
>> 3+2*i
ans =
     7
```

もう一度虚数単位に戻します。sqrt は平方根をとる関数です。
カウンタとして使用することの多いi,j を除いて、定数は変数として使わないようにしましょう。

```
>> i=sqrt(-1)
ans =
        0 + 1.0000i
```

3.2　主な数学関数

以下はよく使われる関数の一覧です。

演算	MATLAB での表現	備考
平方根	y = sqrt(x)	$y=\sqrt{x}$
指数	y = exp(x)	$y=e^x$
対数	y = log(x)	$y=\log_e x$
	y = log10(x)	$y=\log_{10} x$
三角関数	y = sin(x), y = cos(x), y = tan(x), y = asin(x), y = acos(x), y = atan(x)	$\sin x, \cos x, \tan x, \arcsin x, \arccos x, \arctan x$
絶対値	y = abs(x)	$y=\|x\|$
実数部	y = real(x)	$y=Re\{x\}$
虚数部	y = imag(x)	$y=Im\{x\}$
位相角	y = angle(x)	
四捨五入	y = round(x)	
切捨て	y = fix(x)	
剰余	y = rem(x,y)	

複素数の絶対値と位相角を求めます。

```
>> a=1-1*i;
>> abs(a),angle(a)
ans =
    1.4142

ans =
   -0.7854
```

3.3　行列の操作

演算	MATLAB での表現	備考
ベクトルの長さ	*n* = length(*v*)	
行列の行数	*m* = size(*a*,1)	
行列の列数	*n* = size(*a*,2)	
行列の大きさ	[*m*,*n*] = size(*a*)	
数列の作成	*start*:*end* または *start*:*step*:*end*	要素の範囲指定やループの繰り返しの指定にも用いられる
要素が 0 の行列作成	*a* = zeros(*size*)	*size* には大きさを指定
要素が 1 の行列作成	*a* = ones(*size*)	
単位行列作成	*a* = eye(*size*)	
要素が一様乱数の行列作成	*a* = rand(*size*)	0 から 1 までの実数。

上記の引数で、*size* とあるの部分は、行列の大きさを指定します。以下のような指定方法が可能です。

指定	大きさ	例
4	4×4	zeros(4)
[1 4]	1×4	ones([1 4])
[5 5]	5×5	eye([5 5])
5,4	5×4	zeros(5,4)

□ 行列の表現

MATLAB で行列やベクトルを表現するには、中括弧 ([ ]) を使用します。

列ベクトル (3 5 7) を変数 a に代入します。ベクトルや行列の各列は、中括弧 ([ ]) の中に空白 (またはカンマ (,)) で区切って並べます。

```
>> a=[3 5 7]
a =
     3     5     7
```

行ベクトル $\begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix}$ を変数 b に代入します。ベクトルや行列の各行はセミコロン (;) で区切ります。

```
>> b=[2; 4; 6]
b =
     2
     4
     6
```

行列の例です。各列を空白で区切り、各行をセミコロン (;) で区切って中括弧 ([ ]) の中に並べます。

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

行列の各列の数は同じでなくてはなりません。

```
>> a=[2 3 4; 5 6]
??? All rows in the bracketed expression must
have the same number of columns.
```

□ 行列の作成

要素がすべて 0 の 3×2 の行列を作成します。

```
>> x=zeros([3 2])
x =
     0     0
     0     0
     0     0
```

要素がすべて1の行ベクトルを作成します。

```
>> y=ones([1 5])
y =
     1     1     1     1     1
```

0から1までの 2 × 4 の一様乱数を作成します。

```
>> rand(2,4)
ans =
    0.2190    0.9347    0.0346    0.0077
    0.0470    0.3835    0.0535    0.3834
```

10から15までの数列を作成します。

```
>> z=10:15
z =
    10    11    12    13    14    15
```

10 から 2 おきに 20 までの数列を作成します。

```
>> z=10:2:20
z =
    10    12    14    16    18    20
```

20から10までの2づつ減少する数列を作成します。

```
>> z=20:-2:10
z =
    20    18    16    14    12    10
```

このコロン(:)による数列の作成方法は、後述する行列の要素の指定や、ループの繰り返しにも使用されます。

□ 行列の長さ・大きさ

```
>> A=[2 3; 4 5; 6 7];
>> v=[4 -1];
```

ベクトルの長さを調べます。

```
>> length(v)
ans =
     2
```

行列の大きさを調べます。
結果は[行数 列数]の行列で表示されます。

```
>> size(A)
ans =
     3     2

>> size(v)
ans =
     1     2
```

第2引数に1を指定すると行数だけを返します。
同様に2を指定すると列数だけを返します。

```
>> size(A,1)
ans =
     3

>> size(A,2)
ans =
     2
```

右図のように、結果を n と m の2つの変数で受け取ると、n には行数、m には列数を返します。

```
>> [n,m]=size(A)
n =
     3

m =
     2
```

size関数のように、MATLABは、2つ以上の値を返す関数があります。

この場合左辺は、受け取る変数名を中括弧([ ])の中にカンマ(,)で区切って並べます。

□ 行列の参照

行列を操作する場合、例えば、ある範囲の小行列を入れ換えたり、ある行すべてを取り出したり、といった操作を頻繁に行いますが、MATLABではそれらをコロン(:)や中括弧([ ])を使って以下のように簡潔に記述できます。

A(i,j)	i行j列目	A(2:5,3)	2～5行3列目
A(:,j)	j列目	A(2:5,:)	2～5行目
A(i,:)	i行目	A(2:5,1:2)	2～5行1～2列目
A([2 3 4 5],3)	2～5行3列目	A(2:2:7,3)	2,4,6行3列目

行列Aの2～3行の2列目を参照します。

```
>> A(2:3,2)
ans =
     5
     7
```

行列Aの2行目と1行目の、2列目と1列目を取り出します。

```
>> A([2 1],[2 1])
ans =
     5     4
     3     2
```

□ 行列への代入

行列への代入は、右辺と左辺の大きさをそろえる必要があります。

行列Aの2行1列目に0を代入します。

```
>> A(2,1)=0
A =
     2     3
     0     5
     6     7
```

右辺と左辺の大きさが違うとエラーになります。
この様なときは、右辺と左辺の大きさを調べてみてください。

```
>> A(:,2)=v
??? In an assignment  A(:,matrix) = B, the number of rows
in A and B must be the same.

>> size(A(:,2))
ans =
     3     1

>> size(v)
ans =
     2     1
```

左辺の領域を2×1にして代入します。

```
>> A(1:2,2)=v
A =
     2     4
     0    -1
     6     7
```

または、右辺の大きさを3×1にして代入します。

```
>> A(:,2)=[v; 0]
A =
     2     4
     0    -1
     6     0
```

MATLABにおける文字列の扱い

MATLABでは、文字列なども文字コードの行列として扱われます。

```
>> a=['sin curve'];
>> size(a)
ans =
     1     9

>> a(3)
ans =
n
```

3.4　行列(配列)演算

演算	MATLABでの表現	備考
ベクトル(行列)演算	+ - * / ^	
スカラー(配列)演算	+ - .* ./ .^	
行列式	d = det(a)	aは正方行列
転置	a'	複素行列の場合、共役にする
	a.'	複素行列でも共役にしない
逆行列	b = inv(a)	aは正方行列
固有値・固有ベクトル	[v,d] = eig(a)	a*v=v*d
階数	n = rank(a)	

MATLAB は行列演算、配列データ処理に特色があります。また、前出の数学関数を行列(配列データ)に適用することもできます。

□ スカラー積

演算子の前にピリオド(.)を付けると、すべての要素に対する処理になります[2]。

行列 A と3の積(スカラー積)を求めます。

```
>> A.*3
ans =
     6     9
    12    15
    18    21
```

ベクトル $\begin{pmatrix} 2 \\ 3 \end{pmatrix}$ と v とのスカラー積を求めます。

```
>> [2; 3].*v
ans =
     8
    -3
```

□ 転置

A の転置行列を求めます。

${}^tA = \begin{pmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \end{pmatrix}$

```
>> A'
ans =
     2     4     6
     3     5     7
```

[2]MATLAB のマニュアルでは、「行列(Matrix)の操作」と「配列(Array)の操作」を区別しています。ここでの行列とは、数学で扱う狭い意味での行列であり、配列とは、単なる値の並び(データの集合)を意味しています。ピリオド(.)をつけない場合は行列として扱われ、ピリオド(.)を演算子の前につければ配列として扱われることになります。

□ 内積

ベクトル $(2,3)$ と v の内積を求めます。

```
>> [2 3]*v
ans =
     5
```

□ 行列積

行列 A とベクトル v の積を求めます。

```
>> A*v
ans =
     5
    11
    17
```

行列 A と A の転置行列との積を求めます。

```
>> A*A'
ans =
    13    23    33
    23    41    59
    33    59    85

>> A'*A
ans =
    56    68
    68    83
```

3.5　変数の操作

演算	MATLAB での表現	備考
変数の一覧	who	
変数の詳細一覧	whos	
変数の消去	clear *var-name*	
変数のセーブ	save *file-name var-name*	
変数のロード	load *file-name*	

□ 変数の一覧

who コマンドは変数の一覧を表示します。
whos コマンドは更に詳しい変数の一覧を表示します。

```
>> who
Your variables are:

A          a          ans        b          i

>> whos
  Name        Size    Elements    Bytes   Density  Complex

    A         2 by 3         6       48      Full     No
    a         1 by 3         3       24      Full     No
  ans         1 by 1         1        8      Full     No
    b         3 by 1         3       24      Full     No
    i         1 by 1         1        8      Full     No

Grand total is 14 elements using 112 bytes
```

□ 変数の削除

いらなくなった変数は、clear コマンドで削除します。

```
>> who

Your variables are:

A          a          ans        b          i

>> clear a
>> who

Your variables are:

A          ans        b          i
```

clear コマンドを引数なしで実行すると、すべての変数が消去されるので注意してください。

```
>> clear
>> who

Your variables are:
```

3.6　ファイル入出力

演算	MATLAB での表現	備考
変数の書きだし	save *filename varname...*	
	save *filename*	
変数の読み込み	load *filename*	
アスキーデータの読み込み	load *file-name* -ascii	
アスキーデータの書き込み	save *file-name var-name* -ascii	
ファイルの内容の実行	*file-name*	
関数の呼出し	*func-name*(*arg1*,*arg2*,...)	
シェルコマンドの実行	! *shell-command*	

全ての変数をファイルに書きだしておき、次回起動したときに読み込んでやると、前回の状態のまま作業ができます。

MATLAB に式を入力する方法として、直接入力する以外に、あらかじめコマンドをファイルへ入力しておき、そのファイルを読み込み、実行させることもできます。

□ 変数の書きだし／読み込み

すべての変数を書きだして、MATLAB を終了します。
書き出すファイル名を指定しないと、カレントディレクトリの matlab.mat というファイルに書き出されます。

```
>> save

Saving to: matlab.mat

>> quit

 312 flops.
```

MATLABを起動し、変数を読み込みます。
読み込むファイル名を指定しないと、カレントディレクトリの matlab.mat というファイルから読み込まれます。

```
% matlab

        ...

>> who

Your variables are:


>> load

Loading from: matlab.mat

>> who

Your variables are:

A          v
```

□ アスキーデータの読み込み

カレントディレクトリの data.ascii というアスキー形式のデータファイルを読み込みます。
ファイル名(拡張子を除いた)と同じ名前の変数(この場合 data)に格納されます。

```
>> load data.ascii -ascii
>> who

Your variables are:

A          ans        data       v
```

□ シェルの実行

!を先頭に付けると、シェルのコマンドが実行できます。

```
>> !cat data.ascii

    65  5657    -3573
    47  5611    -2994
    57  5547    -2414
    21  5472    -1878
    40  5397    -1314
       ...
```

□ バッチファイル[3]の実行

あらかじめファイルにコマンド列をエディタなどで入力しておき、一括して実行させることができます。その場合、ファイルの拡張子は必ず .m にします。

ファイル innerp.m には、行列 A とベクトル v の内積を計算するコマンド列が入力されています。拡張子を除いたファイル名を指定すると、ファイルの内容が読み込まれ、実行されます。

```
>> !cat innerp.m
A=[2 3; 4 5; 6 7];
v=[4; -1];

A*v
>> innerp
ans =
     5
    11
    17
```

□ 関数の作成

何回も使う処理を新たに関数として作成することができます。

[3]MATLABでは script file と呼んでいます。

バッチファイルの実行 と同様に、拡張子が .m のファイルを作成し、コマンド列を入力しておきます。

バッチファイルと違うのは、ファイルの先頭に関数の仮引数や返値を定義することと、ファイル内で使用された変数はローカルな変数(別のファイルからは参照できない変数)になることです。

実は、今までできた関数は、MATLAB の中にあらかじめ組み込まれて (built-in されて) いるものもありますが、*.m の関数ファイルとして提供されているものもあります。MATLAB のディレクトリの下の toolbox の下にこのような関数ファイルが格納されています。

作成の方法などは、**プログラミング** の項を参照してください。

3.7　プログラミング

演算	MATLAB での表現	備考
条件分岐	if *expr*,*states*,end	
	if *expr*,*states*,else,*states*,end	
	if *expr*,*states*, elseif,*expr*,*states*, else,*states*, end	
繰り返し	for *var*=*expr*,*states*,end	
	while *expr*,*states*,end	
ループからの抜け出し	break	
関数の定義	function	
関数からの復帰	return	

何回も使う処理を新たに関数として定義したり、値の大小など条件によって処理を変えたりすることができます。

MATLAB には、条件分岐や繰り返しを行うif...end, for...end, while...end などの制御文があり、条件判断も<, >, <=, >=, ==, ~=, &, | など csh に近い表現になっています。

□ 制御文

data の中で 3 以上 5 より小さいデータを数えます。

```
>> c=0;
>> for i=1:length(data)
     if (data(i) >= 3) & (data(i) < 5)
       c=c+1;
     end
   end
>> c
ans =
     2
```

□ 関数の定義

上記の処理を関数にしてみます。エディタでファイルに入力します[4]。
ファイル名は、 count3to5.m とします。(関数名は count3to5)

1 行目は関数の宣言です。1 次元配列 data を受け取って、データ数 c を返します。

2 行目はコメントです。% 以降、行末までコメントととして扱われます。しかも、宣言の直後のコメントは、help コマンドの表示として使われます。

4 行目もコメントですが、help コマンドで表示されません。

5 行目は入力引数のチェックをしています。引数の数が 1 より小さい場合、そのままリターンします。

9 〜 13 行目は、ループです。データの要素を 1 つづつ調べて 3 以上 5 より小さい数があったら、カウンタを増やしています。

```
>> !cat count3to5.m
 1 function c=count3to5(data)
 2 %COUNT3TO5 counts more than 3 less than 5.
 3
 4 % check arg
 5 if nargin < 1, return, end
 6
 7 % counting
 8 c=0;
 9 for i=1:length(data)
10   if (data(i) >= 3) & (data(i) < 5)
11     c=c+1;
12   end
13 end
```

[4] 行番号は説明のためにつけてあります。

4　データ処理の応用

4.1　信号処理

演算	MATLABでの表現	備考
フーリエ変換	y = fft(x)	$n = 2^m$ の時 FFT $y = \sum_{r=1}^{n} a_r e^{2\pi i(r-1)(s-1)/n}$
逆フーリエ変換	y = ifft(x)	$y = \frac{1}{n}\sum_{s=1}^{n} b_s e^{-2\pi i(r-1)(s-1)/n}$
2次元フーリエ変換	y = fft2(x)	
2次元逆フーリエ変換	y = ifft2(x)	
コンボリュージョン	y = conv(a,b)	length(y) = length(a)+length(b)-1
2次元コンボリュージョン	y = conv2(a,b)	
デコンボリュージョン	[q,r] = deconv(b,a)	$b = conv(q,a) + r$
デジタルフィルタ	y = filter(b,a,x)	FIR, IIR

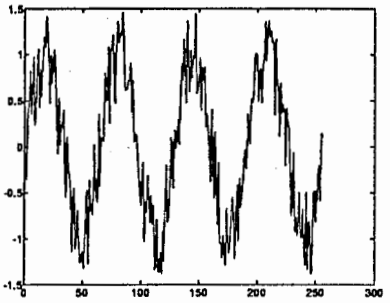
□ 畳み込み

入力信号のサンプルとして、ランダムノイズをのせた正弦波から256点のデータを作成します。

```
>> n=256;
>> rn=rand([1 n]); t=0:1/256:1-1/256;
>> flist=sin((4*2*pi).*t)+(rn-1/2);
```

データをプロットします。

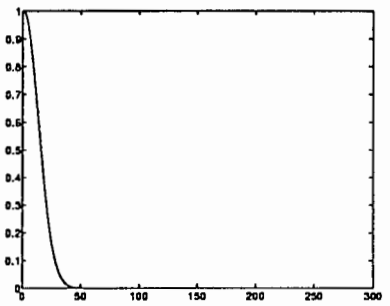
```
>> plot(flist)
```



フィルタ関数として、同様に256点のデータを作成します。

```
>> hlist=exp(-200.*t.^2);
```

```
>> plot(hlist)
```



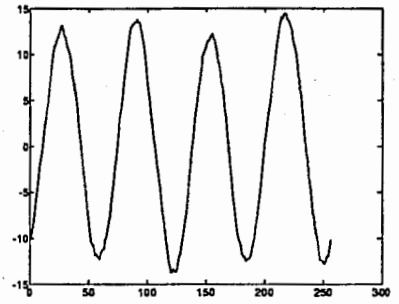
フーリエ変換します。

```
>> fspec=fft(flist);
>> hspec=fft(hlist);
```

fspecとhspecを逆フーリエ変換します。

```
>> ff=real(ifft(fspec.*hspec));
```

```
>> plot(ff)
```



4.2　統計処理

演算	MATLABでの表現	備考
最大値	y = max(x)	
	y = max(x,y)	
最小値	y = min(x)	
	y = min(x,y)	
総和	y = sum(x)	
累積和	y = cumsum(x)	
累積	y = prod(x)	
平均	y = mean(x)	
分散	y = cov(x)	
標準偏差	y = std(x)	
中央値	y = median(x)	
相関係数	y = corrcoef(x)	
正規分布	y = randn($size$)	平均0、分散1
ヒストグラム	y = hist($data$,n)	
	[n,x] = hist($data$,n)	
データの検索	y = find($expr$)	

データが2次元配列の場合、MATLABでは、列優先(column-wise)になります。
したがって和や平均などの計算は列方向に行われます。

□ 平均・分散

各要素が正規分布に従う1次元配列データdata1Dと、2次元配列データdata2Dを作成します。

```
>> data1D=randn(1,10)
ans =
  Columns 1 through 6

   0.8476    0.2681   -0.9235   -0.0705    0.1479   -0.5571

  Columns 7 through 10

  -0.3367    0.4152    1.5578   -2.4443

>> data2D=randn(5,3)
ans =
   -1.0982   -0.9778   -0.5077
    1.1226   -1.0215    0.8853
    0.5817    0.3177   -0.2481
   -0.2714    1.5161   -0.7262
    0.4142    0.7494   -0.4450
```

平均を求めます。2次元配列データでは、各列の中での平均値を返します。

```
>> mean(data1D)
ans =
   -0.1095

>> mean(data2D)
ans =
    0.1498    0.1168   -0.2084
```

全データの平均値を求めます。

```
>> mean(mean(data2D))
ans =
    0.0194
```

max 、 min は引数を2つとると、どちらか大きい方(小さい方)の値を返します。
右記の例では、結果的に0より小さいデータを0に置き換えることができます。

```
>> max(data1D,0)
ans =
  Columns 1 through 7

    0.8476  0.2681       0       0  0.1479       0       0

  Columns 8 through 10

    0.4152  1.5578       0
```

□ データの検索

find関数は、引数の条件を持つデータを検索して、配列のインデックスを返します。
右記の例では、0以上1以下のデータを検索して、その数を求めてます。

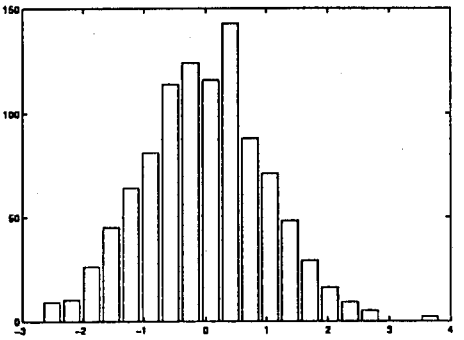
```
>> I=find(data1D>=0 & data1D<=1)
I =
     1     2     5     8

>> length(I)
ans =
     4
```

□ ヒストグラム

正規分布にしたがう1000個のランダムデータのヒストグラムを表示します。最大値と最小値間の分割数は20です。

```
>> data=randn(1,1000);
>> hist(data,20)
```



左辺を指定すると、dataの頻度値、範囲値が得られます。
この場合、グラフはプロットされません。

```
>> [n,h]=hist(data,20);
```

5　数値データのグラフ表示

本章以降では、数値データをグラフ表示する具体例を説明します。
はじめにデータを以下の4つに分類しておきます。

1次元配列データ　1変数の値
2次元配列データ　2変数の格子上の点の値
2次元座標データ　(x, y) が組になったもの
3次元座標データ　(x, y, z) が組になったもの

2次元座標データも3次元座標データも2次元配列データの一種ですが、座標そのものを表示するためのデータという意味で分けておきます。

MATLAB のデータは、1次元の配列または2次元の配列(ベクトルまたは行列)で表されます。本書では、1次元配列とベクトル、2次元配列と行列を同じ意味に使用し、とくに配列(または行列)といった場合、1次元配列やベクトルも含むものとします。

5.1　データファイルの読み込み

アスキー形式のファイルのデータをグラフ表示したい場合は、まず、以下の方法でファイルからデータを読み込み配列変数に代入します。

□ アスキー形式データの読み込み

ファイル foo.ascii に2次元配列データが以下のようにアスキー形式で格納されているとします。

z_{11}	z_{12}	z_{13}	...	z_{1n}
z_{21}	z_{22}	z_{23}	...	z_{2n}
		⋮		
z_{m1}	z_{m2}	z_{m3}	...	z_{mn}

```
>> load foo.ascii -ascii
```

結果として、foo という名前の変数に、m × n 行列として格納されます。

```
>> size(foo)
```

□ アスキー形式データでの書きだし

dataZ という変数が m × n 行列とすると、ファイル foo.ascii に dataZ の値が

z_{11}	z_{12}	z_{13}	...	z_{1n}
z_{21}	z_{22}	z_{23}	...	z_{2n}
		⋮		
z_{m1}	z_{m2}	z_{m3}	...	z_{mn}

のようにアスキー形式で書き出されます。

```
>> save foo.ascii dataZ -ascii
```

5.2　1次元配列データのグラフ表示

data1D = $[x_1, x_2, ...]$ のように1次元配列の値が data1D に代入されているとします。

演算	MATLAB での表現	備考
点グラフ	plot(*data1D*,'.')	
折れ線グラフ	plot(*data1D*,'-')	

□ 点グラフ

引数中の文字列は、色・線種を指定します。

線種には次のものが指定できます。

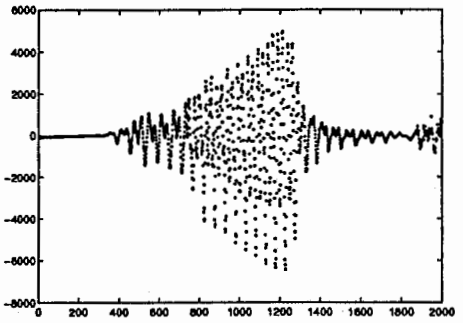
.	点	o	丸
-	線	x	×
:	点線	+	＋
--	鎖線	*	アスタリスク
-.	1点鎖線		

色には以下の文字が指定できます。

y	黄色	m	マゼンダ
c	シアン	r	赤
g	緑	b	青
w	白	k	黒

例えば、黒色の1点鎖線の場合、'k-.'のように指定します。

```
>> plot(data1D,'.')
```

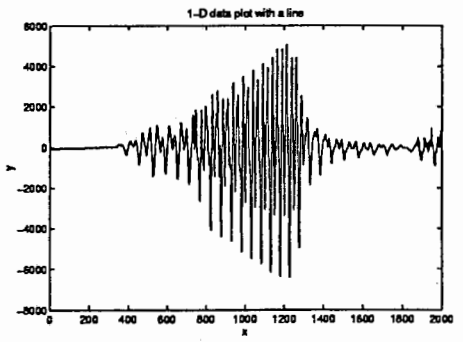


□ 折れ線グラフ

線種を指定しないと、黄色の折れ線グラフになります。

xlabel(str) x軸のラベル。
ylabel(str) y軸のラベル。
title(str) グラフのタイトル。

```
>> plot(data1D)
>> xlabel('x')
>> ylabel('y')
>> title('1-D data plot with a line')
```



5.3　2次元配列データのグラフ表示

dataZ = [$x_{11}, x_{12}, \ldots$; $x_{21}, x_{22}, \ldots$] のように2次元配列の値がdataZに代入されているとします。

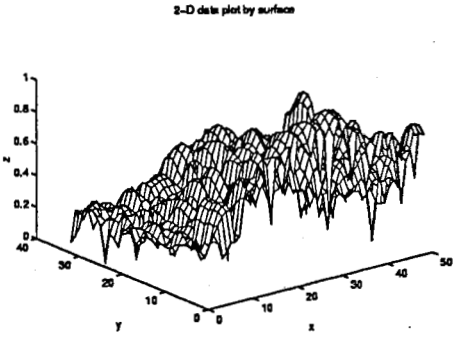
演算	MATLABでの表現	備考
サーフェス表示	surf($dataZ$)	
	surf($x, y, dataZ$)	
等高線表示	contour($dataZ$)	
	contour($x, y, dataZ$)	
濃淡表示	imagesc($dataZ$)	
	imagesc($x, y, dataZ$)	

□ サーフェス表示

サーフェス表示のグラフです。

zlabel(str) z 軸のラベル。
view(az,el) 視線の方向角。

```
>> surf(dataZ)
>> xlabel('x'),ylabel('y'),zlabel('z')
>> title('2-D data plot by surface')
```



□ 等高線表示

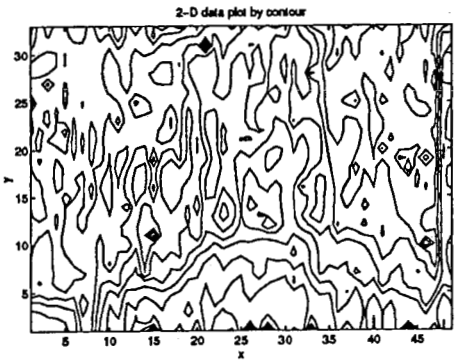
等高線表示のグラフです。
contour(dataZ,N) または
contour(x,y,dataZ,N) でレベル数N を指定できます。

contour(dataZ,V) または
contour(x,y,dataZ,V) でレベルV を指定できます。

contour(dataZ,20) レベル数 20 の等高線

contour(dataZ,[0.4 0.6]) レベル 0.4 と 0.6 の等高線

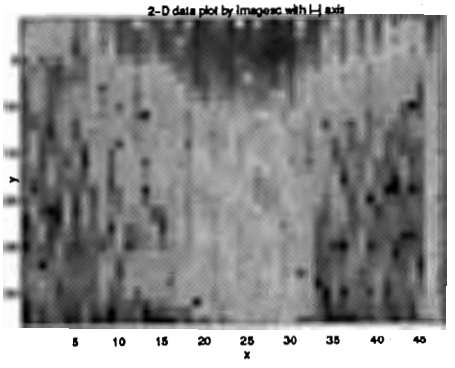
```
>> contour(dataZ)
>> xlabel('x'),ylabel('y')
>> title('2-D data plot by contour')
```



□ 濃淡表示

濃淡表示のグラフです。
y 軸に注意してください。左上が原点となっています。
これは画像データを表示したとき、上下反対にならないように座標系を変えているためです。

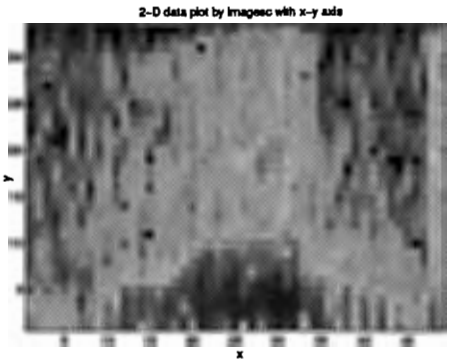
```
>> imagesc(dataZ)
>> xlabel('x'),ylabel('y')
>> title('2-D data plot by imagesc with i-j axis')
```



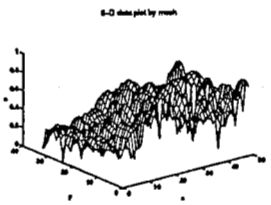
このように、xy 座標系で表示したい場合は、imagesc コマンドの後で、axis コマンドで座標系を指定してやる必要があります。

axis(axis) 座標系の変更。

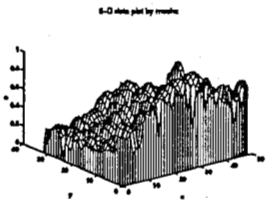
```
>> imagesc(dataZ)
>> axis('xy')
>> xlabel('x'),ylabel('y')
>> title('2-D data plot by imagesc with x-y axis')
```



この他にも メッシュのみ表示する mesh 、サーフェス (メッシュ) と等高線を同時に表示する surfc, meshc、外側をカーテンで囲む surfz, meshz、列方向 (x 方向) の線のみ表示する waterfall などの関数があります。

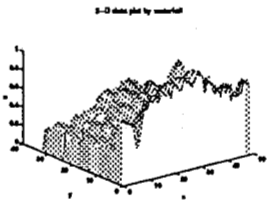


mesh 関数



meshz 関数

meshc 関数



waterfall 関数

5.4　2次元座標点の表示

data2D = [x_1, y_1; x_2, y_2; ...] のように 2 次元座標点が data2D に代入されているとします。

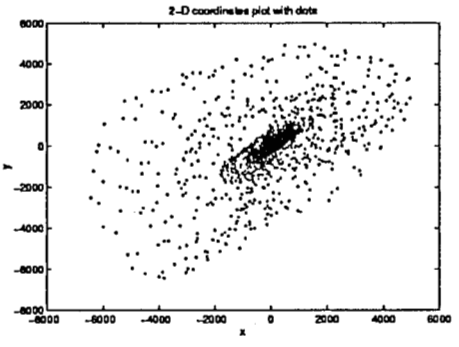
演算	MATLAB での表現	備考
2 次元座標点の点グラフ	plot(x,y,'.')	
2 次元座標点の線グラフ	plot(x,y,'-')	

1 次元配列データと同様、平面上の点や線は plot 関数を使います。

□ 2 次元座標点の点グラフ

配列の 1 列目の x 座標を第 1 引数に、2 列目の y 座標を第 2 引数に指定します。

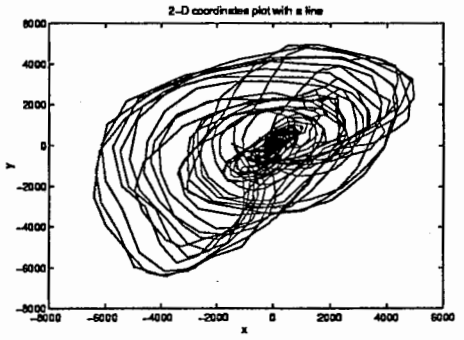
```
>> plot(data2D(:,1),data2D(:,2),'.')
>> xlabel('x'),ylabel('y')
>> title('2-D coordinates plot with dots')
```



□ 2次元座標点の折れ線グラフ

線種を指定しないと、折れ線グラフになります。

```
>> plot(data2D(:,1),data2D(:,2))
>> xlabel('x'),ylabel('y')
>> title('2-D coordinates plot with a line')
```



5.5　3次元座標点の表示

data3D = [x_1, y_1, z_1; x_2, y_2, z_2; ...] のように3次元座標点が data3D に代入されているとします。

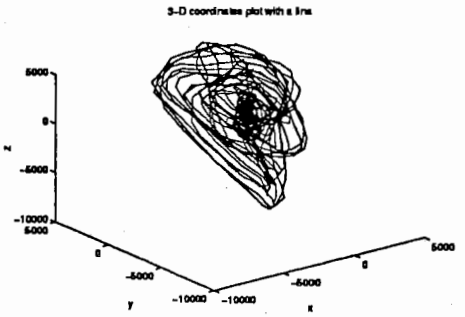
演算	MATLAB での表現	備考
3次元座標点の点グラフ	plot3(x,y,z,'.')	
3次元座標点の線グラフ	plot3(x,y,z,'-')	

3次元空間上の点や線は、plot3 関数を使います。

□ 3次元座標点の折れ線グラフ

配列の 1,2,3 列目の x,y,z 座標を第1,2,3引数にそれぞれ指定します。

```
>> plot3(data3D(:,1),data3D(:,2),data3D(:,3))
```



6　関数のグラフ表示

演算	MATLAB での表現	備考
1変数関数のグラフ	fplot(func,[min max])	
2変数の格子データ作成	[X,Y]=meshgrid(x,y)	

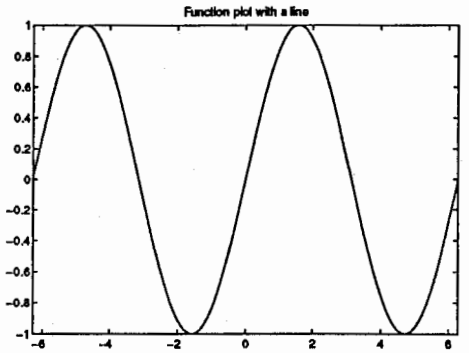
6.1　1変数関数のグラフ表示

□ 1変数関数の折れ線グラフ

fplot 関数は、第1引数に実行するファイル名または関数を文字列で指定、第2引数に定義域の範囲を指定します。[-2, 12] は変数 x の -2π から 2π までの範囲を描画せよという指示です。

右記に挙げた例は、
fplot('sin',[-2*pi 2*pi])
でも同じ結果になります。

```
>> fplot('sin(x)',[-2*pi 2*pi])
```



fplot 関数は、誤差に基づいて x 座標のステップ幅を自動的に計算して表示していますが、右記は単純に概形を得ることができます。

```
>> plot(sin(-2*pi:pi/12:2*pi))
```

6.2　2変数関数のグラフ表示

2変数関数をグラフ表示させる関数は特に用意されてません。しかし2変数から格子上の x 座標データと y 座標データを作成して、要素同士の行列演算を行い、結果を3次元表示することは簡単にできます。

□ サーフェス表示

x 軸上での x 座標 x 、y 軸上での y 座標 y を作成します。

```
>> x=-4:0.2:4;
>> y=-2:0.2:2;
```

格子上の x 座標 X 、y 座標 Y をそれぞれ作成します。x が n 次元ベクトル、y が m 次元ベクトルなら、X,Y は、 m × n 行列となります。

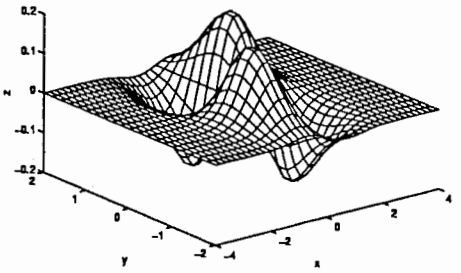
```
>> [X,Y] = meshgrid(x,y);
```

格子上の x 座標 X と y 座標 Y から、行列の要素同士の演算によって、格子上の値 Z を求めます。

```
>> Z = X.*Y.*exp(-X.^2-Y.^2);
```

得られた格子上の値 Z を表示します。

```
>> surf(x,y,Z)
```



7　複数のグラフ表示およびその他のグラフ表示

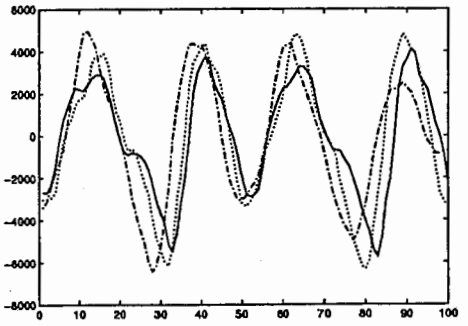
7.1　複数のグラフ表示

演算	MATLAB での表現	備考
複数のグラフを重ねて表示	hold on	
複数のグラフを並べて表示	subplot(row, col, no)	
ウィンドウのクリア	clf	(CLear Figure)

□ 複数のグラフを重ねて表示

配列の 1001～1100 までのデータを実線、1101～1200 までを点線、1201～1300 までを一点鎖線で重ねて表示します。
hold on コマンドは上書きモードにし、座標軸の目盛りを固定します。hold off コマンドは逆に書き換えモードにし、座標軸の目盛りを自動的につけかえるようにします。

```
>> plot(data1D(1001:1100),'-')
>> hold on
>> plot(data1D(1101:1200),':')
>> plot(data1D(1201:1300),'-.')
>> hold off
```



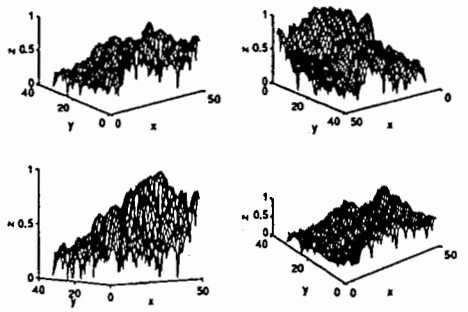
□ 複数のグラフを並べて表示

視線の方向を変えたいくつかの 3 次元グラフを並べて表示します。
subplot 関数は、ウィンドウを分割して複数の座標軸を作成する関数です。
引数には、行方向の分割数、列方向の分割数、何番目か、の 3 つを指定します。
view 関数は、引数なしで実行すると、現在の視線の方位角 az と仰角 el を返します。

```
>> subplot(2,2,1)
>> mesh(dataZ)
>> xlabel('x'),ylabel('y'),zlabel('z')
>> [az,el]=view;
```

同じグラフを視線の方位角、仰角を変えて表示します。

```
>> subplot(2,2,2)
>> mesh(dataZ)
>> xlabel('x'),ylabel('y'),zlabel('z')
>> view(az-180,el)
>>
>> subplot(2,2,3)
>> mesh(dataZ)
>> xlabel('x'),ylabel('y'),zlabel('z')
>> view(az,5)
>>
>> subplot(2,2,4)
>> mesh(dataZ)
>> xlabel('x'),ylabel('y'),zlabel('z')
>> view(az,60)
```



7.2 その他のグラフ表示

演算	MATLAB での表現	備考
1次元配列データの片対数グラフ	`semilogx(`*data1D*`) or semilogy(`*data1D*`)`	
1次元配列データの両対数グラフ	`loglog(`*data1D*`)`	
棒グラフ	`bar(`*data1D*`)`	

7.3 グラフの出力

演算	MATLAB での表現	備考
ファイルへの出力	`print -d`*device filename*	
プリンタへの出力	`print -d`*device* `-P`*printer*	

引数 *device* にはフォーマットを指定します。デフォルトでは、PostScript 形式ですが、GIF 形式や各社プリンタのフォーマットで出力することができます。詳しくは、print コマンドのヘルプを参照してください。

□ プリンタへの出力

現在のウィンドウを PostScript 形式でプリンタ lw へ出力します。

```
>> print -Plw
```

□ ファイルへの出力

プリンタではなくファイルに出力する場合もプリンタへの出力と同様です。

現在のウィンドウを PostScript 形式でファイルへ出力します。
この場合ファイル名は、foo.ps になります。

```
>> print foo
```

8 グラフィックスの基礎

演算	MATLAB での表現	備考
新たなウィンドウの作成	*obj*`=figure`	
新たな座標系の作成	*obj*`=axes`	
現在のウィンドウの変更	`figure(`*obj*`)`	
現在の座標系の変更	`axes(`*obj*`)`	
属性値の設定／変更	`set(`*obj,prop-name,prop-value,...*`)`	
属性値の表示／取得	*prop-value*`=get(`*obj,prop-name*`)`	

これまでは、2次元や3次元のデータを簡単に視覚化する方法を説明してきました。しかし、座標軸の目盛りの設定や、x軸の表示範囲の変更など、より細かい設定を行うためには、MATLAB でグラフィックスはどのように扱われているのかを知っておく必要があります。

MATLAB ではウィンドウや座標軸、2次元／3次元グラフなどのグラフィックスをひっくるめて**グラフィック・オブジェクト**と呼んでいます。そして、座標軸の目盛りやグラフの線種などの設定や変更は、**グラフィック・オブジェクトの属性 (property)** を変更することで可能となります。

基本的に MATLAB のグラフィックスは、これらのオブジェクトを作成する関数 (figure,axes,line,text,etc...) と、属性を参照／設定する関数 (set/get) の組合せで、グラフィックスの作成や細かい設定を行うようになっています。

8.1 グラフィック・オブジェクトの種類と作成

まず、グラフィック・オブジェクトの種類には、以下のものがあります。

名前／関数	説明	他の作成関数
figure	ウィンドウ。	
axes	座標系。ウィンドウ内の1つの図に対応する。	subplot
line	線図形。	plot, plot3
surface	3次元サーフェス。	surf, mesh
image	2次元イメージ。	imagesc
text	文字列。	
patch	多角形のパッチ。	waterfall
uicontrol	GUI[5]のボタン、スライダなどの部品。	
uimenu	GUIのメニュー。	

オブジェクト名とオブジェクトを作成する関数名は同じです。これらの関数は、オブジェクトのハンドル (識別番号、ID) を返します。

□ オブジェクトの作成と属性の参照

新たな figure を作成して、axes を作り、line を1つ作成します。
それぞれのオブジェクトを作成する関数は、オブジェクトのハンドルを返します。

```
>> h1=figure
h1 =
     2

>> h2=axes
h2 =
   63.0001

>> h3=line
h3 =
   68.0001
```

それぞれのオブジェクトが持っている属性とその属性の値を調べます。

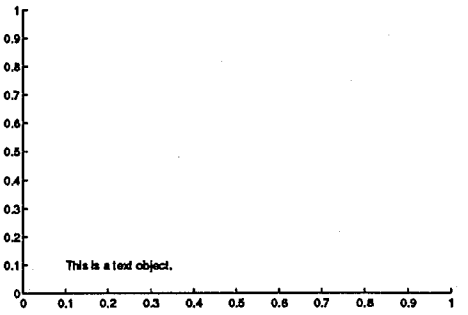
```
>> get(h1)
        BackingStore = on
        Color = [0 0 0]
        Colormap = [ (64 by 3) ]
        CurrentAxes = [63.0001]
             ...
>> get(h2)
        AspectRatio = [NaN NaN]
        Box = off
        CLim = [0 1]
        CLimMode = auto
             ...
>> get(h3)
        Color = [1 1 1]
        EraseMode = normal
        LineStyle = -
        LineWidth = [0.5]
             ...
```

[5] グラフィック・ユーザ・インタフェース (Graphic User Interface)。

□ 文字列の表示

文字列を表示するには、text オブジェクトを作成します。
text 関数は、引数に文字列の位置と、文字列そのものを指定することができます。

```
>> h=text(0,0,'This is a text object.');
```



get 関数で、オブジェクトがどんな属性を持っているかを調べることができますが、属性の値の型がいくつかあり、どんな値をとるかは属性によって決まっています。
付録に オブジェクトの属性一覧をつけてありますので、詳しくは付録を参照してください。

8.2 グラフィック・オブジェクトの属性の設定 / 変更

□ 位置の変更

axes を作成し位置を変更します。
まず長さの単位 (Units) を設定し、ウィンドウの位置 (Position) に、[x y 幅 高さ] のベクトルを設定します。
右記では、'normalized' と指定することによりウィンドウの左下を (0,0)、右上を (1,1) とする正規化された単位で位置・大きさを指定しています。

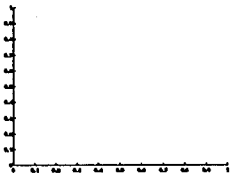
なお、2行目の ... はコマンドや式を次の行へ続ける場合に使用します。

```
>> h=axes;
>> set(h,'Units','normalized',...
>>   'Position',[.1 .1 .5 .5])
```

□ 座標軸の範囲の変更

axes を作成します。作成したオブジェクトのハンドル (識別番号) を返します。

```
>> obj=axes
obj =
   65.0004
```



作成された axes の属性の一覧を表示します。

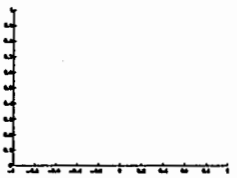
```
>> get(obj)
        AspectRatio = [NaN NaN]
        Box = off
        CLim = [0 1]
        CLimMode = auto
        Color = none
           ...
```

指定した属性の値を取得します。ここでは属性として x 軸の範囲を受け取ります。関数は [*min max*] のベクトルを返します。

```
>> xlim=get(obj,'XLim')
xlim =
     0     1
```

取得した属性の値を変更します。ここでは、x 軸の最小値を -1 に変更しています。

```
>> set(obj,'XLim',[-1 xlim(2)])
```



9 Q&A

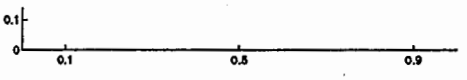
グラフの目盛りを変えるには？

グラフの目盛りは MATLAB が自動的に付けてくれますが、

□ 目盛りの数値をかえる場合

x 軸の目盛りを 0.1,0.5,0.9 にします。

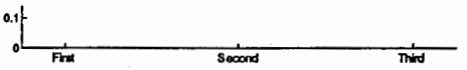
```
>> set(gca,'XTick',[0.1 0.5 0.9])
```



□ 目盛りの文字列をかえる場合

x 軸の目盛り文字列を変更します。ここでは、目盛りの数が 3 つなので、3 行の文字列配列で指定します。

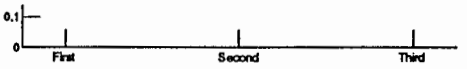
```
>> set(gca,'XTickLabels',['First ';'Second';'Third '])
```



□ 目盛りの長さをかえる場合

目盛りの長さを変更しています。
ただし、x,y 軸ともに変更されます。

```
>> set(gca,'TickLength',[0.04 0.02])
```

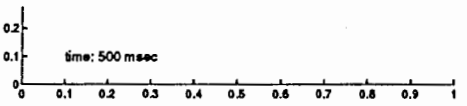


文字の大きさを変えるには？

□ text オブジェクトの大きさを変える場合

text オブジェクトを作成して、そのフォントの大きさを 14pt に変更しています。

```
>> h=text(0.1,0.1,'time: 500 msec');
>> set(h,'FontSize',14)
```



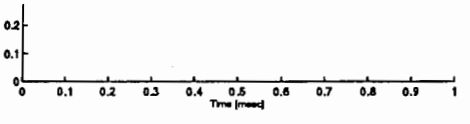
なお作成コマンドの引数で設定することもできます。

```
>> text(0.1,0.1,'time: 500 msec','FontSize',14)
```

□ x,y 軸のラベルやタイトルの大きさを変える場合

x 軸のラベルの文字の大きさを 10pt に設定します。
まず x 軸のラベル値は、text オブジェクトのハンドルになっていますから、その text オブジェクトのフォントの大きさを変更します。

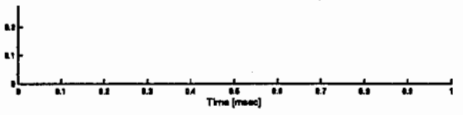
```
>> h=get(gca,'XLabel');
>> set(h,'FontSize',10)
```



□ x,y 軸の目盛り文字列の大きさを変える場合

x,y 軸の目盛り文字列の大きさは、座標軸のフォントの大きさの属性 (FontSize) で変更できます。

```
>> set(gca,'FontSize',8)
```



ウィンドウの大きさを変えるには？

現在のウィンドウの左下の位置を (10, 10)、幅 600、高さ 400 に変更します。単位は、'pixel' です。

```
>> set(gcf,'Units','pixel',...
>>   'Position',[10 10 600 400])
```

上記の設定は、現在のウィンドウだけ有効ですが、以後作成されるすべてのウィンドウに設定したい場合は、右記のように、ルートウィンドウ (オブジェクト・ハンドルが 0) に、デフォルト値として設定します。

```
>> set(0,'DefaultFigureUnits','pixel',...
>>   'DefaultFigurePosition',[10 10 600 400])
```

グラフィックスを TeX に取り込むには？

print コマンドで -d オプションに eps を指定すると、eps 形式[6]で出力されます。

グラフィックスを eps 形式でファイルへ出力します。

```
>> print -deps foo
```

eps 形式のファイルを TeX に取り込む方法は、dvi 形式から PostScript 形式へ変換するドライバによって異なりますが、右記に dvips の場合の例をあげておきます。

```
\documentstyle[epsf,...
        .
        .
\epsfxsize=10cm
\epsfbox{foo.eps}
        .
        .
```

プリンタへの出力の位置・大きさを変えるには？

a4 用紙の横方向に一杯の大きさで出力します。

```
>> set(gcf,'PaperType','a4letter',...      % 紙の大きさ
>>   'PaperOrientation','landscape',...    % 紙の方向
>>   'PaperUnits','centimeter',...         % 数値の単位
>>   'PaperPosition',[0.8 0.8 19 28])      % 紙への出力位置
```

上記の設定は、現在のウィンドウだけ有効ですが、以後作成されるすべてのウィンドウに設定したい場合は、右記のように、ルートウィンドウ (オブジェクト・ハンドルが 0) に、デフォルト値として設定します。
MATLAB を起動するたびに設定するのが面倒な場合は、startup.m ファイルに書いておきましょう。

```
>> set(0,'DefaultFigurePaperType','a4letter',...
>>   'DefaultFigurePaperOrientation','landscape',...
>>   'DefaultFigurePaperUnits','centimeter',...
>>   'DefaultFigurePaperPosition',[0.8 0.8 19 28])
```

謝辞

本編の作成にあたっては、第六研究室の佐藤 雅昭氏に全面的に御協力いただきました。
ここに記して感謝します。

[6] Enhancement PostScript。大きさが変更できる PostScript 形式。

付録： グラフィック・オブジェクトの属性一覧

figure の属性一覧

属性名	機能	属性値
BackingStore	バッキングストア機能	{'on'} | 'off'
Color	ウィンドウ内の色	色指定*または 'none'
Colormap	カラーマップ	$m \times 3$ 行列
CurrentAxes	操作対象の axes オブジェクト	axes オブジェクトのハンドル
CurrentObject	操作対象のオブジェクト	オブジェクトのハンドル
InvertHardcopy	白黒逆転ハードコピー	'on' | 'off'
KeyPressFcn	キーボード押下時の関数	文字列
MenuBar	メニューバー	'none' | {'figure'}
MinColormap	最小カラー数	整数
Name	ウィンドウ名	文字列
NextPlot	次の描画モード	'new' | {'add'} | 'replace'
NumberTitle	番号タイトル表示 / 非表示	{'on'} | off
PaperUnits	プリント用紙の大きさの単位	{'inches'} | 'centimeters' | 'normalized' | 'points'
PaperOrientation	プリント用紙の方向	{'portrait'} | 'landscape'
PaperPosition	プリント用紙の位置	4 次元ベクトル
PaperType	プリント用紙の形	{'usletter'} | 'uslegal' | 'a3' | 'a4letter' | 'a5' | 'b4' | 'tabloid'
Pointer	マウスポインタの形	'crosshair' | {'arrow'} | 'watch' | 'topl' | 'topr' | 'botl' | 'botr' | 'circle' | 'cross' | 'fleur'
Position	ウィンドウの位置	位置指定†
Resize	大きさの変更	{'on'} | 'off'
ShareColors	色の共有	'no' | {'yes'}
Units	ウィンドウの大きさの単位	'inches' | 'centimeters' | 'normalized' | 'points' | {'pixels'}
WindowButtonDownFcn	マウスボタン押下時の関数	文字列
WindowButtonMotionFcn	マウスボタン移動時の関数	文字列
WindowButtonUpFcn	マウスボタンリリース時の関数	文字列
ButtonDownFcn	マウスボタン押下時の関数	文字列
Clipping	クリッピング	{'on'} | 'off'
Interruptible	割り込み許可	{'no'} | 'yes'
Parent	親オブジェクト	ルートウィンドウのハンドル(0)
UserData	ユーザデータ	どんなデータでも可
Visible	表示 / 非表示	{'on'} | 'off'

* 色指定: 3 次元ベクトル ($[red\ green\ blue]$ $0 \leq red, green, blue \leq 1$) または 文字列 ('y'|'c'|'g'|'w'|'m'|'r'|'b'|'k')

† 位置指定: 4 次元ベクトル ($[x\ y\ width\ height]$)

axes の属性一覧

属性名	機能	属性値
AspectRatio	座標軸のサイズ比	2 次元ベクトル
Box	座標系の枠	'on' | {'off'}
CLim	色座標の範囲	2 次元ベクトル
CLimMode	色座標の範囲モード	{'auto'} | 'manual'
Color	座標系の色	色指定*または 'none'
ColorOrder	色の順番	$m \times 3$ 行列
DrawMode	描画速度モード	{'normal'} | 'fast'
FontAngle	フォントの傾き	{'normal'} | 'italic' | 'oblique'
FontName	フォント名	文字列
FontSize	フォントサイズ	数値
FontWeight	フォントの濃淡	'light' | {'normal'} | 'demi' | 'bold'
GridLineStyle	グリッドの線種	'-' | '--' | {':'} | '-.'
LineStyleOrder	線種の順番	文字列配列
LineWidth	線の太さ	数値
NextPlot	次の描画モード	'new' | 'add' | {'replace'}

属性名	機能	属性値
Position	座標系の位置	位置指定†
TickLength	目盛りの長さ	2 次元ベクトル
TickDir	目盛りの方向	{'in'} | 'out'
Title	タイトル	text オブジェクトのハンドル
Units	長さの単位	'inches' | 'centimeters' | {'normalized'} | 'points' | 'pixels'
View	視線の方角	2 次元ベクトル
XColor	x 軸の色	色指定*
XDir	x 軸の方向	'normal' | {'reverse'}
Xform	投射行列	4 × 4 行列
XGrid	x 軸のグリッド	'on' | {'off'}
XLabel	x 軸のラベル	text オブジェクトのハンドル
XLim	x 軸の範囲	2 次元ベクトル
XLimMode	x 軸の範囲設定モード	{'auto'} | 'manual'
XScale	x 軸のスケール	{'linear'} | 'log'
XTick	x 軸の目盛り	n 次元ベクトル
XTickLabels	x 軸の目盛り文字列	文字列配列
XTickLabelMode	x 軸の目盛り文字列設定モード	{'auto'} | 'manual'
XTickMode	x 軸の目盛り設定モード	{'auto'} | 'manual'
YColor	y 軸の色	色指定*
YDir	y 軸の方向	{'normal'} | 'reverse'
YGrid	y 軸のグリッド	'on' | {'off'}
YLabel	y 軸のラベル	text オブジェクトのハンドル
YLim	y 軸の範囲	2 次元ベクトル
YLimMode	y 軸の範囲設定モード	{'auto'} | 'manual'
YScale	y 軸のスケール	{'linear'} | 'log'
YTick	y 軸の目盛り	n 次元ベクトル
YTickLabels	y 軸の目盛り文字列	文字列配列
YTickLabelMode	y 軸の目盛り文字列設定モード	{'auto'} | 'manual'
YTickMode	y 軸の目盛り設定モード	{'auto'} | 'manual'
ZColor	z 軸の色	色指定*
ZDir	z 軸の方向	{'normal'} | 'reverse'
ZGrid	z 軸のグリッド	'on' | {'off'}
ZLabel	z 軸のラベル	text オブジェクトのハンドル
ZLim	z 軸の範囲	2 次元ベクトル
ZLimMode	z 軸の範囲設定モード	{'auto'} | 'manual'
ZScale	z 軸のスケール	{'linear'} | 'log'
ZTick	z 軸の目盛り	n 次元ベクトル
ZTickLabels	z 軸の目盛り文字列	文字列配列
ZTickLabelMode	z 軸の目盛り文字列設定モード	{'auto'} | 'manual'
ZTickMode	z 軸の目盛り設定モード	{'auto'} | 'manual'
ButtonDownFcn	マウスボタン押下時の関数	文字列
Clipping	クリッピング	{'on'} | 'off'
Interruptible	割り込み許可	{'no'} | 'yes'
Parent	親オブジェクト	figure オブジェクトのハンドル
UserData	ユーザデータ	どんなデータでも可
Visible	表示 / 非表示	{'on'} | 'off'

†位置指定: 4 次元ベクトル ($[x\ y\ width\ height]$)

*色指定: 3 次元ベクトル ($[red\ green\ blue]$ $0 \leq red, green, blue \leq 1$) または 文字列 ('y'|'c'|'g'|'w'|'m'|'r'|'b'|'k')

line の属性一覧

属性名	機能	属性値
Color	色	色指定*
EraseMode	消去モード	{'normal'} | 'background' | 'xor' | 'none'
LineStyle	線種	{'-'} | '--' | ':' | '-.' | '+' | 'o' | '*' | '.' | 'x'
LineWidth	線の太さ	数値
MarkerSize	マーカーの大きさ	数値

属性名	機能	属性値
Xdata	x データ	n 次元ベクトル
Ydata	y データ	m 次元ベクトル
Zdata	z データ	$m \times n$ 行列
ButtonDownFcn	マウスボタン押下時の関数	文字列
Clipping	クリッピング	{'on'} | 'off'
Interruptible	割り込み許可	{'no'} | 'yes'
Parent	親オブジェクト	axes オブジェクトのハンドル
UserData	ユーザデータ	どんなデータでも可
Visible	表示 / 非表示	{'on'} | 'off'

＊色指定: 3 次元ベクトル ($[red\ green\ blue]$ $0 \leq red, green, blue \leq 1$) または 文字列 ('y'|'c'|'g'|'w'|'m'|'r'|'b'|'k')

text の属性一覧

属性名	機能	属性値
Color	文字の色	色指定*
EraseMode	消去モード	{'normal'} | 'background' | 'xor' | 'none'
FontAngle	フォントの傾き	{'normal'} | 'italic' | 'oblique'
FontName	フォント名	文字列
FontSize	フォントサイズ	数値
FontWeight	フォントの濃淡	'light' | {'normal'} | 'demi' | 'bold'
HorizontalAlignment	水平方向の割り付け	{'left'} | 'center' | 'right'
Position	位置	位置指定†
Rotation	回転角度	数値
String	文字列	文字列
Units	長さの単位	'inches' | 'centimeters' | 'normalized' | 'points' | 'pixels' | {'data'}
VerticalAlignment	鉛直方向の割り付け	'top' | 'cap' | {'middle'} | 'baseline' | 'bottom'
ButtonDownFcn	マウスボタン押下時の関数	文字列
Clipping	クリッピング	{'on'} | 'off'
Interruptible	割り込み許可	{'no'} | 'yes'
Parent	親オブジェクト	axes オブジェクトのハンドル
UserData	ユーザデータ	どんなデータでも可
Visible	表示 / 非表示	{'on'} | 'off'

＊色指定: 3 次元ベクトル ($[red\ green\ blue]$ $0 \leq red, green, blue \leq 1$) または 文字列 ('y'|'c'|'g'|'w'|'m'|'r'|'b'|'k')
†位置指定: 4 次元ベクトル ($[x\ y\ width\ height]$)

uicontrol の属性一覧

属性名	機能	属性値
BackgroundColor	背景色	色指定*
CallBack	コールバック関数	文字列
ForegroundColor	前景色	色指定*
HorizontalAlignment	水平方向の割り付け	'left' | {'center'} | 'right'
Max	最大値	数値
Min	最小値	数値
Position	位置	位置指定†
String	文字列	文字列
Style	部品の種類	{'pushbutton'} | 'radiobutton' | 'checkbox' | 'edit' | 'text' | 'slider' | 'frame' | 'popupmenu'
Units	長さの単位	'inches' | 'centimeters' | 'normalized' | 'points' | {'pixels'}
Value	値	数値
ButtonDownFcn	マウスボタン押下時の関数	文字列
Clipping	クリッピング	{'on'} | 'off'
Interruptible	割り込み許可	{'no'} | 'yes'
Parent	親オブジェクト	figure オブジェクトのハンドル
UserData	ユーザデータ	どんなデータでも可
Visible	表示 / 非表示	{'on'} | 'off'