MATLAB 簡易マニュアル

中野倫靖[†]

作成日 2003.09.02

改訂日 2003.12.07

[†]**筑波大学 図書館情報メディア研究科**
nakano@slis.tsukuba.ac.jp

1. MATLAB* とは [1)]

MATLAB は対話型の数値計算ソフトウェアであり，特に行列計算に強みがある．

C 言語や Fortran を用いた伝統的な数値計算と比べ，MATLAB を使うメリットとしては，以下のような点が挙げられる．

- 行列演算が高速
- コーディングが容易
- データ構造に気をとらわれない（配列を使うのに宣言がいらない，など）
- 対話型のインタフェースにより，高速な実験と容易なデバッグが可能
- 高品質なグラフィックとビジュアル化機能が使用可能
- MATLAB の M-file は，広範囲のプラットフォーム対応が可能
- Toolbox を追加し，システムを拡張することが可能
- インターネットには多くの M-file が公開されており，自由に利用可能

また，MATLAB はインタプリタ言語であり，コンパイル形式の言語と比べると効率が悪い点がある．しかし，これは MATLAB コンパイラを使用したり，MEX ファイルを用いて Fortran や C とリンクさせたりすることによって解決できる．

MATLAB では，容易に高品質の処理が行えるので，研究で数値計算（グラフィックス面）や信号処理，数式処理を行う人は知っておいて損はない．

2. 筑波大学春日キャンパスで MATLAB

春日キャンパスの MATLAB は MATLAB Student Edition Ver.5.3 であり，インストールされている PC は，「情報処理実験・演習室 I」の「ma1－ma5」と「md1－md5」の計 10 台である．また，研究用として，大学内の LAN に接続されている PC なら，そこへインストールして使うことができる†. インストール CD は長谷川先生が管理しているので，必要がある人は長谷川先生（hasegawa@slis.tsukuba.ac.jp）に相談してほしい．

注意点として，ライセンスの関係上，MATLAB の同時実行が最大で 4 人までなので，MATLAB を起動させたままという状況は避けるべきである．

2.1. ライセンスの構成

- MATLAB 4
- MATLAB Web Server 1
- Symbolic Math Toolbox 4
- Signal Processing Toolbox 2
- Wavelet Toolbox ?

---

* MATLAB は MathWorks の登録商標である．

† インストールマニュアルを参照．http://＊＊＊（現在は未設置）

3. MATLAB の基礎

3.1. 実行画面

MATLAB を実行すると図 1 のような画面になる.

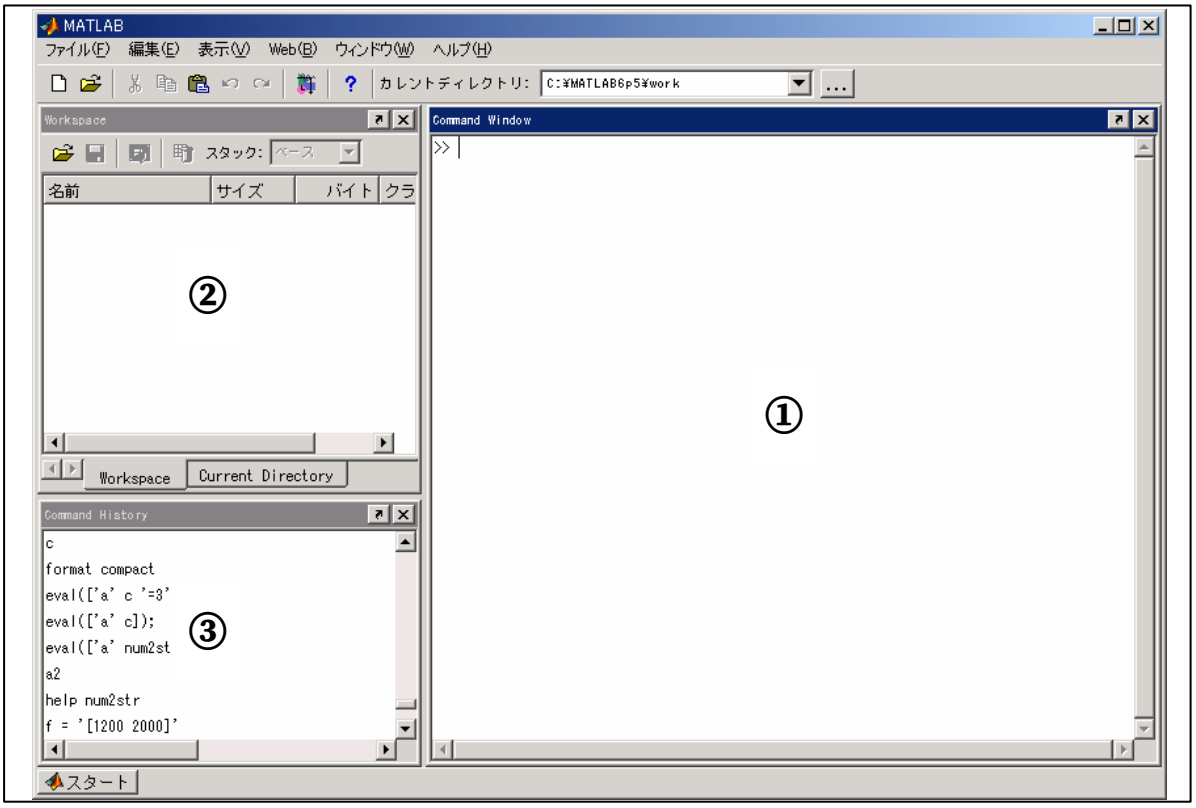


図 1　MATLAB 実行画面

画面構成は以下のようになる.

① コマンドウィンドウ
② ワークスペース
③ コマンドヒストリ

主に使用するのはコマンドウィンドウであり，ここで入出力を行う．また，ワークスペースは，使用している変数の一覧が表示される場所，コマンドヒストリは，過去に打ったコマンドの一覧が表示される.

3.2. 演算

MATLAB での計算には，表 1 のような演算子を用いる*.

表 1　MATLAB 演算子

演算	演算子	比較演算	演算子	ベクトル 行列	演算子
和	＋	等しい	= =	ベクトル	:
差	ー	等しくない	~=	転置	’
積	* (.*)	小なり	<		
商（右割）	/ (./)	大なり	>		
左割†	\ (.\)	以下	<=		
累乗	^ (.^)	以上	>=		
		論理積	&		
		論理和	|		

簡単な計算例を示す．“ >> ”は，プロンプトを示す．

```
>> 1+2
ans =
     3
>> 76-23
ans =
    53
>> 4*8
ans =
    32
>> 8/2
ans =
     4
>> 2^10
ans =
        1024
```

* 括弧内の演算は，行列ベクトル演算特有のものである．“ 3.5 行列演算 ”で説明する．
† バックスラッシュは，日本語キーボードでは “ ¥ ”記号となる．

3.3. 変数

MATLAB の計算には変数を用いることができる．大文字と小文字は区別されるので，注意してほしい．計算例を以下に示す．

```
>> a = 7                ①
a =
       7
>> b = 8*9;             ②
>> b                    ③
b =
     7 2
>> a-b                  ④
ans =
    -65
>> ans                  ⑤
ans =
    -65
>> c = [1:3]            ⑥
c =
      1    2    3
```

```
>> d = c'                    ⑦
d =
       1
       2
       3
>> e = [10:-2:1]             ⑧
e =
     1 0    8    6    4    2
>> f = [1 2 3; 4,5,6]        ⑨
f =
       1     2     3
       4     5     6
>> f(1,2)                    ⑩
ans =
       2
>> e(2:4)                    ⑪
ans =
       8    6    4
```

① 変数 a に 7 を代入している．“ = （イコール）” は代入を表し，「等しい」という意味の演算子は，表 1 に示すように “ == （ダブルイコール）” である．

② b=8*9 で最後にセミコロン “ ; ” がついているが，セミコロンをつけると計算結果を出力しない．変数の値を出力するには③のようにする．

④ 結果を変数に代入しなかった場合，“ ans ” という名の変数が自動的に割り当てられる．⑤は ans 内容の表示．

⑥ 1 次元行ベクトルを作っている．“ 1:3 ” は，「1 から 3 まで 1 ずつ増加させる」という意味で，“ [] ” がベクトルを表す．⑧のように “ [10:-2:1] ” とすると，「10 から 1 まで 2 ずつ減少させたベクトルを作れ」という演算になる．

⑦ 転置により列ベクトルを作成している．

⑨ 行列を作る演算である．データは，半角スペースかカンマ “ , ” で区切る．行の区切りは，例のように “ ; ” か，その場で改行する．

⑩ 行列要素へのアクセスである．MATLAB では，ベクトル・行列の添え字は 1 から始まる．f(1,2)で，行列 f の 1 行 2 列の要素を表す（丸括弧であることに注意）．また，⑪のようにすると e の 2～4 の要素，という意味になる．

3.4. 右割と左割

MATLAB には，通常の商を表す右割に加えて，左割があり，以 下のような働きをする．

$$Right \quad division : a / b \qquad \frac{a}{b}$$

$$Left \quad division : a \setminus b \qquad \frac{b}{a}$$

この左割は，主に行列演算に使用する．すなわち，A,B,X を行列としたとき，A/B は X*B=A の X を求め，A¥B によって A*X=B の X を求める．実際にやってみよう．

```
>> A=[1 2; 3 4], B = eye(2)        ①
A =
     1     2
     3     4
B =
     1     0
     0     1
>> X=A/B                           ②
X =
     1     2
     3     4
>> X*B                             ③
ans =
     1     2
     3     4
```

```
>> A=[1 2; 3 4], B = eye(2)        ④
A =
     1     2
     3     4
B =
     1     0
     0     1
>> X = A¥B                         ⑤
X =
   -2.0000    1.0000
    1.5000   -0.5000
>> A*X                             ⑥
ans =
    1.0000         0
    0.0000    1.0000
```

① “ , ”は 1 行に複数の式を書く場合に用いる記号, eye(2)は単位行列を作る MATLAB の関数である．

② 右割を用い，X*B=A の X を求めている．結果の確認を③で行っており，A と値が一致しているのが分かるだろう．

④ ①と同じ

⑤ 左割を用い，A*X=B の X を求めている．結果の確認は⑥で行っている．

3.5. 行列同士の演算

行列（ベクトル）同士では，特殊な演算がある．例えば，行列同士の積といった場合，普通，以下のようになる（変数 A,B は **3.4.**の値を用いた）.

```
 >> A*B
ans =
     1     2
     3     4
```

行列の各要素同士の演算の場合，演算子の前に“ . ”をつける．そうすると，各要素同士の積を要素に持つ行列を求めるには，以下のようになる.

```
>> A.*B
ans =
     1     0
     0     4
```

積と同様に，商（右割,左割）・累乗にも“ . ”をつけることが可能である.

3.5.1. 行列作成関数

3.4.で出てきた eye と同じ使い方をするものに，zeros，ones がある．ones は全要素が 1 の行列，zeros は全要素が 0 の零行列，eye は対角要素が 1 で他の要素が 0 の単位行列を, それぞれ作成する．zeros(a)や zeros(a,b)のように, 引数は 1 つもしくは 2 つである．引数が 1 つの場合は，正方行列，2 つの場合は，a×b の行列が作成される.

良く使うベクトル作成関数として，linspace，logspace*がある．linspase は，等間隔なベクトルを，logspace は対数的に等間隔なベクトルを作成する.

```
>> x = linspace(1,10,5)
x =
   1.0000   3.2500   5.5000   7.7500  10.0000
>> y= logspace(1,3,3)
y=
   10      100      1000
```

* linspace(a,b,c)は a から b まで c 点を作成．logspace(a,b,c)は 10^a から 10^b まで c 点を作成.

3.6. help

MATLAB 関数の説明などを，自分で調べる場合は help コマンドを用い，“ >> help *function_name* ”のようにする．また，単に help と打つと，以下のような出力を得る*.

```
>> help
HELP トピック:
matlab¥general      - 一般的なコマンド
matlab¥ops          - 演算子と特殊キャラクタ
matlab¥elmat        - 基本行列と行列操作
matlab¥elfun        - 初等数学関数
matlab¥matfun       - 行列関数- 線形数値代数
matlab¥graph2d      - 2 次元グラフ
matlab¥graph3d      - 3 次元グラフ
matlab¥iofun        - ファイルの入出力
```

この結果を基に，さらに help を繰り返すことによって，求める関数の説明を得る．

```
>> help graph2d
  2 次元グラフ
  基本的な X-Y グラフ
  plot      - 線形プロット
  loglog    - 両対数スケールプロット

>> help plot
  PLOT    線形プロット
  PLOT(X,Y) は、ベクトル X に対してベクトル Y をプロットします……
```

GUI でヘルプを見るには，以下のコマンドを用いる．

```
>> helpdesk
```

また，キーワード *string* を基に，関数の検索を行いたい場合は以下のようにする．

```
>> lookfor string
```

* ここでは，省略して表示している．

3.7. 高速化

MATLAB は同じ演算でも，計算方法によって実行速度が大きく違う．結論から言えば，MATLAB の組み込み関数を用いるのが最も高速である．

例えば，ベクトルの要素を全て足し合わせる，という演算の場合，for 文を用いる方法と，組み込み関数の sum を用いる方法とがある．

```
>> a = [1:10000];
>> tic, tmp = 0;, for i=1:10000, tmp = tmp + a(i);, end, toc
elapsed_time =
    0.0500
>> tic, sum(a);, toc
elapsed_time =
    0.0200
```

例を見ても分かるように，sum 関数を使った方が，実行時間が短い*．扱うベクトルがもっと大きい場合（音楽を扱うなら 1 秒で 44100 点）や，何度も処理を繰り返すような場合は，組み込み関数を使うべきである．それによって実行時間を削減でき，コードもすっきりする．

また，行列の要素に対する演算の場合にも MATLAB らしいコーディングがある．例えば，行列 A の各要素と行列 B の各要素同士の積を得たい場合，①for 文を使う方法と，②“ .* ”を用いる方法がある．もちろん，②が MATLAB らしい．

```
>> A = eye(1,1000);
>> B = 3*ones(1,1000);
>> for i=1:1000, C(i) = A(i) * B(i);, end, C;          ①
>> D = A.* B;                                          ②
```

ここで用いた例はデータ数が少なく，計測時間に違いが見られなかったので割愛する†．ただ，コードが見やすくなっているのは分かるだろう．

また，for 文を使わなければ実現できない，という状況もあるだろう．そういった場合は，まず，zeros 関数などで，必要な大きさの行列を作成し，メモリを確保しておく．そうすると，上の for 文の例のような場合でも，比較的高速に処理できる．これは，①の前に“ >> C = zeros(1,10) ”という行をおくことに相当する．

---

* tic（計測開始），toc（計測終了），である．

† データ数を 10000 などと大きくすれば違いが出る．試して欲しい．

3.8. グラフィックス

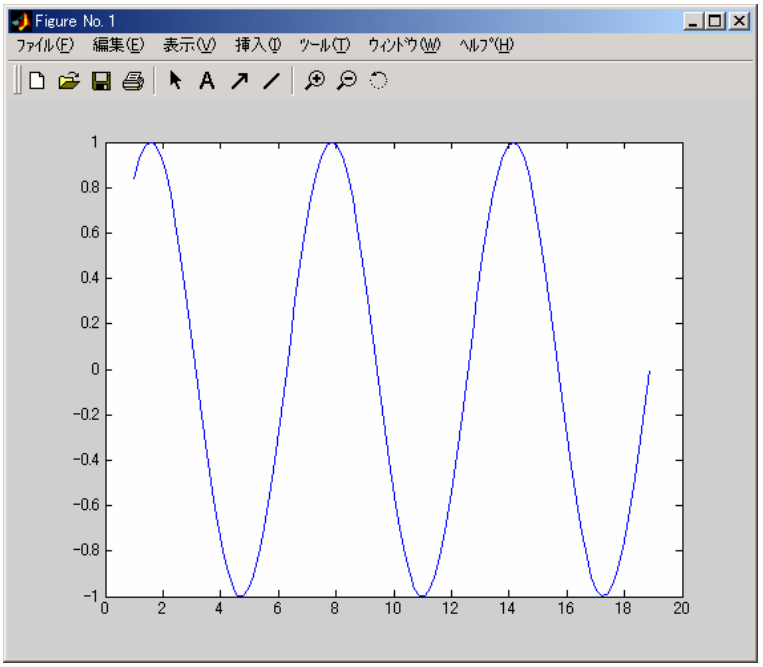
代表的なグラフィックス関数に plot 関数がある．

plot(y)とすると，ベクトル y をグラフに描画する．この時，横軸は点の番号となる．

plot(x,y)とすると，ベクトル x に対応してベクトル y を描画する．この時，x と y は同じ次元（長さ）でなければならない．

plot 関数は，いくつかのプロパティを持つ．例えば，デフォルトの plot(x)では青い実線を描画するが，plot(x,'r:')とすると赤い点線の描画を行う．これらの詳しい解説は，参考文献や help で得てほしい．

```
>> x = linspace(1,6*pi,100);
>> y = sin(x);
>> plot(x,y)
```



グラフ描画関数は，実行すると「Figure No.1」というウィンドウを生成し，そこに描画する．いくつかグラフを作るとき，何もしないとグラフは上書きされてしまうので，figure 関数を用いて，figure ウィンドウを制御する．“ figure(2) ”のように用いると，新たに「Figure No.2」のウィンドウが生成され，その次にグラフ描画を行えば，No.2 のウィンドウにグラフが描写される．

次のグラフは Figure No.2 のウィンドウに描画される．これをカレントフィギュアという．figure 関数の引数には正の整数を与える．既に作成されている figure 番号を指定することもできる．

3.9. M-file

MATLAB を用いたプログラミングでは M-file を用いる．M-file は拡張子‘ .m ’を持つテキストファイルで，MATLAB のコマンドが収められており，二通りの使用法がある．

i. コマンドの列を一気に実行する（スクリプト M-file）

ii. 新しい関数を作成する（ファンクション M-file）

M-file を作成するには，メニューバーのファイルから M-file を選択すればよい．作成した M-file を実行するには，そのファイル名をコマンドラインで打てばよい．例えば，“ test.m ”という M-file を作成したなら*,“ >>test ”とコマンドを打つ†.

i は，まとまった処理を行う場合に用いる．

ii の用法としては，例えば，以下のような M-file“ sample.m ”を作成する．

```
function sample(x)
% M-file のサンプルです.
% sample(x)とすると，ベクトル x をプロットします.


figure(1)
plot(x)     %  ベクトル x のプロット
```

ここで,“ % ”はコメントを表す．function 行のすぐ下のコメントは,“ >>help sample ”と打ったときに表示するコメントを表す．この関数 sample は x を引数として受け取り，plot 関数によって描画を行う．引数は，複数設定することも可能である．

また，返り値（出力）を設定することも可能で，その場合，以下のようにする．

```
function out = sample2(in,pow)
% sample2(in, pow)のように使います.
% in の pow 乗を出力します.


out = in^pow;
```

結果は，以下のようになる．

```
>> sample2(3,3)
ans =
    27
```

* M-file 名が既存の関数名と重複しないように注意.

† この場合，カレントディレクトリに M-file がある必要がある．ディレクトリの移動は，ディレクトリウィンドウで行う．

3.9.1. Flow Control

プログラミング上，流れを制御する関数には，if, for, while, switch があり，その書式，及び使用例は，以下のようになる．

[if 文]

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

※“ elseif ”は，“ else ”と“ if ”の間にスペースが入らないことに注意．

（使用例）

```
if(x > y)                    % ( )はあってもなくても良い
    temp = y
else
    temp = x
end
```

[for 文]

```
for variable = expression
    statements
end
```

（使用例 1）

```
for x = [pi/6 pi/4 pi/3]     % 式はベクトルで与える
    y = sin(x)
end
```

（使用例 2）

```
for i = 1:10          % start : step : end という書式（:step は省略可）
    j = i^2
end
```

[while 文]

```
while expression
    statements
end
```

（使用例）

```
x = 1;
while x > 0
    xmin = x;
    x = x/2;
end
xmin
```

この例では，数（double）の最小値を求めている．

[switch 文]

```
switch expression
    case expression statements
    otherwise expression statements
end
```

（使用例）

```
x = input('Enter Real Number: ');
switch x
    case {-Inf, Inf}                %Inf.-Inf,NaN の説明
        disp('Plus or Minus Infinity')
    case 0
        disp('Zero')
    otherwise
        disp('Nonzero and Finite')
end
```

input **関数：テキストを出力し，ユーザの入力を待つ関数．**
disp **関数：テキストを出力する関数．**
{ }**：セル配列．異なるタイプの変数（配列，長さの異なる文字列）を格納できる．**
※case **実行後は** swicth **を抜けるので，**C **言語とは違って** break **は必要ない．**

4. MATLAB で信号処理（基礎）

ここでは, 主に WAVE ファイルを読込み, それを処理する方法について簡単に述べる．

WAVE ファイルの読込には，wavread 関数を用いる*.

```
>> [y,Fs,Bits] = wavread('sample.wav');
```

関数によっては，出力引数の数を選ぶこともできる．詳細は各関数の help を参照してほしい．この例の場合は，y にサンプリングされたデータ，Fs にサンプリング周波数，Bits に量子化 bit 数が格納される．y は，振幅が[-1,1]に正規化された，n 行 ch 列の行列となり，n はデータ数，ch はチャンネル数である．

続いて，この波形をグラフにプロットする．

```
>> plot(y);
```

波形を再生するには，sound 関数を用いる．

```
>> sound(y,Fs,Bits);
```

続いて，この信号に対して FFT を施して表示する．この際，1 チャンネルのみを使用する．FFT のサイズは 4096 点とする．

```
>> y1 = y(:,1);            % 1ch 全て．“:”が全ての要素を表す．
>> Y = fft(y1, 4096);      % y1 を 4096 点で fft して Y に格納．
>> Y = Y(1:2048);          % 4096 点のうち，意味があるのは半分の 2048 点である．
>> f = (0:2048-1);         % 横軸のスケール用変数 f.
>> f = f / 4096 *Fs;       % Hz に変換．
>> plot(f, abs(Y))         % Y は複素数なので abs 関数で絶対値を取って表示する．
>> title('Fourier Transform')   % グラフにタイトルをつける
>> xlabel('Freqency[Hz]')       % x 軸のラベル
>> ylabel('Amp')                % y 軸のラベル   グラフィックスの章で説明
```

* WAVE ファイルは概して大きいファイルであるので，セミコロンを付け忘れると，表示に時間がかかることがある．wavread 関数は読込区間を指定することもできるので，それを活用するのも良い．

4.1. 時間周波数解析

信号処理において，周波数解析を時間とともに行い，周波数スペクトルにどのような変化があるかを見ることは非常に重要である．このようなスペクトルの時間変化のパターンを表したものを，スペクトログラムという．

スペクトログラムを表示させるには，Signal Processing Toolbox の specgram 関数を用いる．この時，あまり長いデータ列を与えると時間がかかるので注意すること．

```
>> specgram(y, 4096, Fs, hanning(4096), 4096 - floor(0.01*Fs))
```

ここで第二引数の 4096 は窓長である．窓とは，解析区間の長さのことである．

第三引数の Fs は，wavread によって得たサンプリング周波数である．

第四引数の hanning(4096)は，窓関数に 4096 点のハニング窓を使用することを示す．hanning 関数も Signal Processing Toolbox の関数である．

第四引数 4096 – floor(0.01*fs)は，オーバーラップさせる点の数を表す．つまり，10ms ごとに窓をずらしてゆくことに相当する．floor 関数は，小数点以下を切り上げる関数で，これをつけないと，サンプリング周波数が 22.05kHz などの時にエラーが出てしまう．

4.2. 軸の操作

x 軸や y 軸の表示させる範囲を変更するには，以下の関数を用いる．

x 軸&y 軸変更

```
axis([x-min x-max y-min y-max])
```

x 軸変更

```
xlim([x-min x-max])
```

y 軸変更

```
ylim([y-min y-max])
```

参考文献

[1] Desmond J. Higham; Nicholas J. Higham. “MATLAB guide”. SIAM, 283p, 2000.

[2] 上坂吉則．MATLAB プログラミング入門．星雲社，281p，2000．

関連情報

http://www.slis.tsukuba.ac.jp/~hasegawa/MATLAB