Matlab の使い方

2014 年 1 月

上智大学理工学部情報理工学科

荒井隆行

1. はじめに

Matlab はもともと行列演算を気軽に行えるアプリケーションとして誕生しましたが、その後、様々な点が強化され、今では行列演算はもとより、グラッフィクス機能や GUI (Graphical User Interface)等まで充実したパッケージとなりました。また、オプションとして用意されている Toolbox を合わせて使うことにより、様々な専門分野への応用も可能となりました。その分野は、機械工学、電気・電子工学はもとより、数学・物理学や情報学、また経済学・統計学・心理学など文系の分野に至るまでと、幅広く範囲が広がっています。

本書は、様々なユーザに対する Matlab の入門書として、基本的な操作を重点的に説明します。また、後半以降ではより深く使いたい人のために多少専門的な使い方についても触れます。そこでは、それぞれの目的に合わせた具体的な実行例などについても随時、サンプルプログラムを示していきます。

なお、本書を通じて行頭にある「>>」の記号は Matlab 使用時のプロンプト、すなわちコマンド入力のための促進・待機を示す記号を表します。つまり、読者のみなさんは、この記号「>>」より右側の文字列を実際に Matlab のウィンドウに入力してください。

また、Matlab にはオンラインヘルプが準備されています。もしわからないことがあれば、「help」コマンドを使って、常に調べることができます。

2. はじめの一歩

本章では、まずはじめにすべての基本となる行列の操作の仕方から説明します。

2.1 Matlab の起動

本学内のコンピュータシステムにおいて、Matlab は UNIX と呼ばれる OS（オペレーティングシステム）上で動くようになっています。そのため、その起動の仕方や使用する上での手順において、他の Windows 用のアプリケーションを使う場合よりも若干の知識を要します。しかし、簡便に Matlab を利用するには学内の Windows マシン上で UNIX のための端末ウィンドウを用意し、その中で使用することでほとんどの作業を行えます。

このように、Windows マシン上で UNIX 端末ウィンドウを起動する方法は大きく分けて 2 つあります。1 つは文字端末である「Telnet」を起動する方法、もう１つはグラフィックスも使える「X-Window」のための端末ウィンドウを起動する方法です。

本学の PC 上では、「Telnet」と「X-Window」はともに「Network Connections」というフォルダの中に見つけることができます。いずれかの方法で、UNIX マシンにログインして下さい。ログインに際しては、ログイン名とパスワードを聞いてきますので、普段使っているものを入力してください。できましたら（必要とあれば希望のフォルダ＝ディレクトリに移動した後に）「matlab」＋Enter キーを入力して下さい。しばらくしてから、Matlab の起動メッセージが画面に現れ、プロンプトである「>>」の記号が現れれば準備完了です。（X-Window の場合、ウィンドウの位置を決定する必要があるので、マウスのボタンをクリックしてください。）

図 1 に、X-Window でログインした場合の最初の画面を示します。

2.2 データの入力

まず手始めに、

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

を入力してみて下さい。記号「＝」は右側の右辺の内容を左側の左辺に入力する、という働きをします。左辺の「x」は、いわばデータを保存する箱、つまり変数の意味を持ちます。上記の命令文では、変数 x に右辺を代入する、ということになります。右辺のかぎ括弧で囲われた中身はいわば「データ」を意味します。そして周りの囲っている「かぎ括弧」は

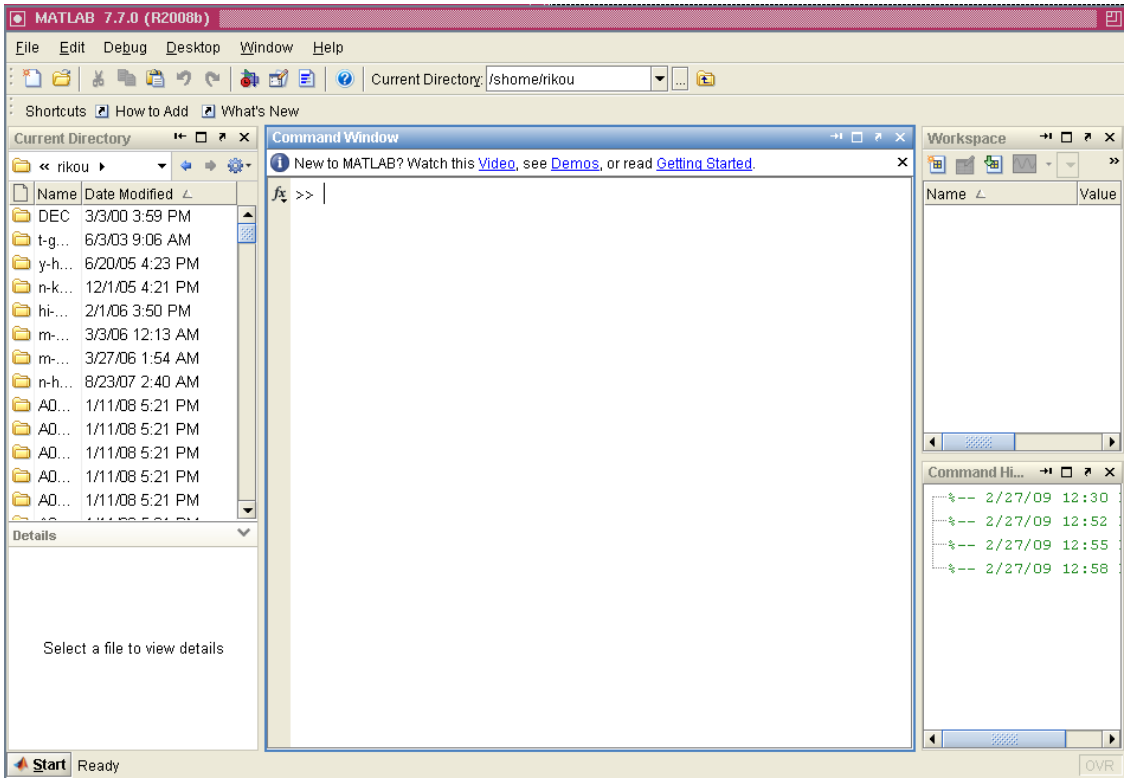

図 1：X-Window でログインした場合の最初の Matlab の画面

それらの要素を「ひとまとまりにする」という働きを持っています。すなわち、かぎ括弧でくくられた全体は、数学でいう行列やベクトルという単位になるわけです。上記の命令を実行すると、横に続くデータ列を形成されます。ここでは、それを「行ベクトル」と呼ぶことにします。この場合、各要素は半角スペースで区切られていますが、カンマを用いることも可能です。

ところで、上記の命令を実行すると、Matlab は x の中身をエコーバック（画面に結果を表示）します。このようにエコーバックしないようにするためには、

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

というように、ステートメント（命令文）の最後にセミコロン「;」を付けます。なお、入力する記号は常に半角文字を使用してください。

ところで、x の中身を見るにはどうしたらいいでしょうか。それには、

```
>> x
```

とだけ入力してください。Enter キーを押すと、中身が画面に出力されます。

2.3 コロン演算子

上記の例のように 3 つの要素だけならばすべてを並べればいいですが、もし 1 から 20 までの整数を要素に持つような行ベクトルを定義しようとして

```
>> x = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
```

を実行したのでは大変なことになります。そこで、Matlab には便利な演算子が用意されています。

```
>> x = [1:20]
```

これを実行することによって、1 から 20 までの整数を順番に要素にもつベクトルが定義されます。

それでは、20 未満の奇数だけを並べたい場合には

```
>> x = [1:2:19]
```

とします。2 つのコロンに挟まれた「2」は、作られる数列の刻み幅（等差数列でいうところの公差）を表します。つまり、これは

```
>> x = [1 3 5 7 9 11 13 15 17 19]
```

と同じ結果を生み出すことがわかります。

2.4 総和を計算する

```
>> x = [1 2 3];
>> sum(x)
```

を実行すると、答として「6」がエコーバックされます。これは、各要素の総和です。このように、Matlab には「sum」に代表されるように様々な関数（括弧の中に指定された入力

に何らかの操作を加え、それを返すサブプログラム）が準備されています。

関数sumは総和を計算しましたが、この他に平均を計算する関数meanなど統計用の関数や、数学で用いられる正弦関数の sin など様々なものが用意されています。

2.5 各要素を取り出す

ベクトル x の i 番目の要素は、x(i)によって取り出すことができます。すなわち、

```
>> x(1)
```

を実行すると、結果として「1」がエコーバックされます。同様に、x(2)の結果は「2」、x(3)の結果は「3」となります。

2.6 行列の入力

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

を実行することによって、行列 A が定義されます。ここで、行列 A の第 1 行・第 1 列の要素は 1、第 1 行・第 2 列の要素は 2、第 1 行・第 3 列の要素は 3、第 2 行・第 1 列の要素は 4…と続きます。「＝」よりも右側の右辺の部分はかぎ括弧[]によって囲われていますが、これは囲われた全体が行列を形作ることを意味しています。またこの場合、行列の中にある記号の「;」は次の行に移るための区切り文字の役目を果たしています。（「;」を用いる代わりに、Enter キーを使って行を変えることによっても入力することが可能です。）このように行列の中で用いられる「;」と、さきほど見た行末にある「;」では役割が違うので注意してください。

2.7 行列の各要素を取り出す

ベクトルの時と同様に、行列 A の i 行 j 列の要素は、A(i,j)によって取り出すことができる。すなわち、

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

を実行すると、結果として「5」がエコーバックされます。

2.8 転置：行列の行と列を入れ替える

次に、

```
>> A'
```

を実行してみましょう。すると、行列 A が転置、すなわち行と列が入れ替わってエコーバックされるのがわかります。つまり、行列にシングルクォート「'」を付与することによって、転置が実行されるのです。

行ベクトルに対しては、

```
>> x'
```

を実行するとわかるように、列ベクトルになります。つまり、もともと行ベクトルであった x が、行と列を入れ替えた結果、（横ではなく）縦に続く数列になったのです。

Matlab で列ベクトルを作る１つの方法は、例えば

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

のようになります。しかし、

```
>> [1 2 3]'
```

というように転置を用いても、同様の結果となることがわかります。

2.9 行列に対して sum を実行すると...

```
>> sum(A)
```

を実行すると、答として「12 15 18」がエコーバックされます。これは、各列ごとに総和した結果です。このように Matlab では、もともとベクトルに対して用意されている関数を行列に対して適用した場合、各列ごとに「sum」が実行され、各列ごとの総和が新たな行ベクトルを形成します。

以上の知識をうまく利用すると、各行ごとの総和を求めることも比較てき容易にできます。

```
>> sum(A')'
```

を実行すると、転置された後の行列に対し列ごとに総和が計算されるので、結果として、Aの行ごとの総和である「6 15 24」がエコーバックされます。

2.10 各行・各列を取り出す

行列Aの第2行を取り出すには

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

と実行します。このように、コロン「:」は該当する全要素を意味する。つまり、

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

と同じことです。第3列を取り出すには、A(:,3)とします。また、

```
>> A(:,end)
```

を実行しても同じ結果が得られると思います。これは、行列Aの最終列を表示せよという意味であり、行列Aは3列しかないので、結局、A(:,3)と同じ意味になるからです。

2.11 行列の一部の値を変更する

行列のある要素だけを更新したい場合、

```
>> A(1,2) = 0
```

などとします。また、行列の行だけ、列だけを入れ替えることもできます。

```
>> A(1,:) = [10 20 30]'
```

を実行すると、１行目だけに代入が行われます。また、

```
>> A(1:2,1:2) = [10 20; 40 50]
```

を実行することによって、行列の一部を（大きさが小さい）別の行列で上書きしてしまう

ことも可能です。

3. Matlab における表現

本章では、Matlab で用いられる 4 つの表現である変数、数値、演算子、関数の 4 つを見ていきます。

3.1 変数

Matlab では変数の型宣言やその大きさの宣言などをしなくてもプログラムを書くことができ、Matlab が自動的にその辺りの割り振りを行ってくれるので、意識せずにプログラミングができるという利点があります。

変数名は通常、最初の 1 文字は数字以外の文字、2 文字目以降は数字を含む文字から構成されています。変数名では大文字小文字の区別がされ、最初の 63 文字が用いられます。

3.2 数値

数値を表現するには、「1」「-2」「300」などのように通常、我々が数値を書く表記法を用いる他、「4.56e7」というように浮動小数点に基づく表記を用いることも可能です。

複素数を表現するには、虚数単位として Matlab ではあらかじめ「i」もしくは「j」が用意されています。つまり、「1i」「2j」などというように虚数を表現します。

3.3 演算子

Matlab で用意されている主な演算子は以下の通りです。

+	足し算
-	引き算
*	掛け算
/	割り算
^	べき乗
'	転置
()	演算を優先的に行うための括弧

２つの行列に対する加減算を実行する場合、行列の大きさが一致している必要があります。

```
>> A = [1 2; 3 4];
>> B = [1 0; 0 1];
>> C = A+B;
```

また、掛け算を実行する際、「*」の左側の行列の列の数が右側の行列の行の数に一致している必要があります。

```
>> A = [1 2 3; 4 5 6];
>> B = [1 0 0 0; 0 1 0 0; 0 0 1 0];
>> C = A*B;
```

ただし、スカラー倍の場合には、単に

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> C = 2*A;
```

などとすればよく、これらの記述法は数学における書き方に似ています。

ところで、大きさが同じ2つの行列の対応する要素同士を掛け算したい場合もあると思います。そのような場合は、ピリオドを以下のように用います。

```
>> A = [1 2; 3 4];
>> B = [2 3; 4 5];
>> C = A.*B;
```

これを実行すると、各要素同士の掛け算が実行されます。つまり、行列Cの第1行第1列の要素は、行列Aの第1行第1列の要素である「1」と行列Bの第1行第1列の要素である「2」の積、行列Cの第1行第2列は、「2」と「3」の積、というようになります。

3.4 関数

第2章で扱った「sum」も関数の例でした。その他、初等数学のための関数に至るまで、Matlabでは様々な関数が用意されています。それらは例えば、以下の通りです。

sin	sin関数
cos	cos関数
tan	tan関数
asin	sinの逆関数
acos	cosの逆関数
atan	tanの逆関数

atan2　atan2(y,x)を実行することで座標(x,y)に対する角度を－π～πの間で返す
abs　絶対値
angle　位相角
sqrt　平方根
real　実部
imag　虚部
conj　複素共役
round　丸め
fix　丸め（0に向かって）
floor　丸め（－∞に向かって）
ceil　丸め（∞に向かって）
sign　引数が正なら1を、負ならば-1を、0ならば0を返す
rem　rem(x,y)を実行することでx/yの割り算の余りを返す
exp　指数関数
log　自然対数
log10　常用対数

[特殊関数]
以下のものはMatlabで用意されている特殊関数ですが、実際は定数として用います。

pi　円周率
iやj　虚数単位
eps　非常に小さな数
Inf　無限大
NaN　Not-a-Number

これらの関数名は、上書きして別の変数名として用いることも可能です。つまり、以下の説明する「for」文ではよく繰り返しを制御するための変数名として「i」や「j」を用いますが、

```
>> for i=1:10
```

などと記述した時点で、特殊関数iは上書きされてしまうことになります。なお、もとの特殊関数に戻すためには、

```
>> clear i
```

を実行します。

3.5 その他

空の行列を作るには、

```
>> A = []
```

を実行します。size 関数は行列の大きさを調べるのにとても有効です。

```
>> size(A)
```

と実行することによって、空の行列であることが確認されます。

4. 制御命令

他のプログラミング言語同様、Matlab には制御命令があります。それらは、if 文、switch 文、for ループ、while ループ、そして break 文です。本章では、それらのうちでもっとも使用頻度の高い if 文と for ループの使い方を見ていきます。

4.1 if 文

if 文ではある論理表現が「真」であるか「偽」であるかを判断します。もし「真」であればある処理をして、「偽」であれば他の処理をする、というようにプログラムの流れを制御します。

例えば、

```
>> if age < 10
     price = 0
   elseif age < 20
     price = 1000
   elseif age < 60
     price = 2000
   else
     price = 1000
   end
```

というような例では、変数 age によって料金のための変数である price の値を設定しています。つまり、10 歳未満はただ、10 歳以上 20 歳未満は 1000 円、20 歳以上 60 歳未満は 2000 円、それ以上は 1000 円といった具合です。

なお、数値の大小関係を調べるには、

```
<       より小さい
<=      以下
>       より大きい
>=      以上
==      等しい
~=      等しくない
```

を用います。また、行列が等しいかを調べるには、「isequal(A,B)」を用います。

4.2 for ループ

for ループは、ある決まった命令を複数回行いたい場合に威力を発揮します。例えば、関数「sum」と同じようにベクトル x の各要素の総和を求める記述を for ループを使って書くとしたら、以下のようになります。

```
>> total = 0;
>> for n = 1:length(x)
     total = total + x(n);
   end
>> total
```

これによって、「sum(x)」と同じ結果が得られることが確認できます。

5. 簡単なグラフィックス

X-Window でログインした場合、グラフィックスを画面上に表示することができます。この章では、グラフィックスの例として、plot 関数を用いた簡単な例を紹介します。

5.1 ２次元プロット

```
>> plot(x)
```

を実行すると、x の中に入っているデータによる 1 本の折れ線が描画されます。ここで、縦軸は各要素の値であるのに対し、横軸は 1, 2, 3, ...というように各要素に対応した自然数が割り振られています。

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> plot(A)
```

というように行列に対して plot を実行すると、3 本の折れ線が描画されます。各列に対応するデータがそれぞれ線で結ばれている様子がわかります。

5.2 放物線のプロット例

ここで簡単な例として、放物線をプロットしてみましょう。横軸として x に-1 から 1 までの値を 0.1 刻みで代入します。そのためには、コロン演算子を使って

```
>> x = [-1:0.1:1];
```

次に、縦軸として y には横軸の値を 2 乗したものを代入します。

```
>> y = x.^2;
```

このように両軸のデータを用意しておき、

```
>> plot(x, y)
```

を実行すると、放物線が折れ線で表現されたプロットが得られるのがわかります（図 2)。

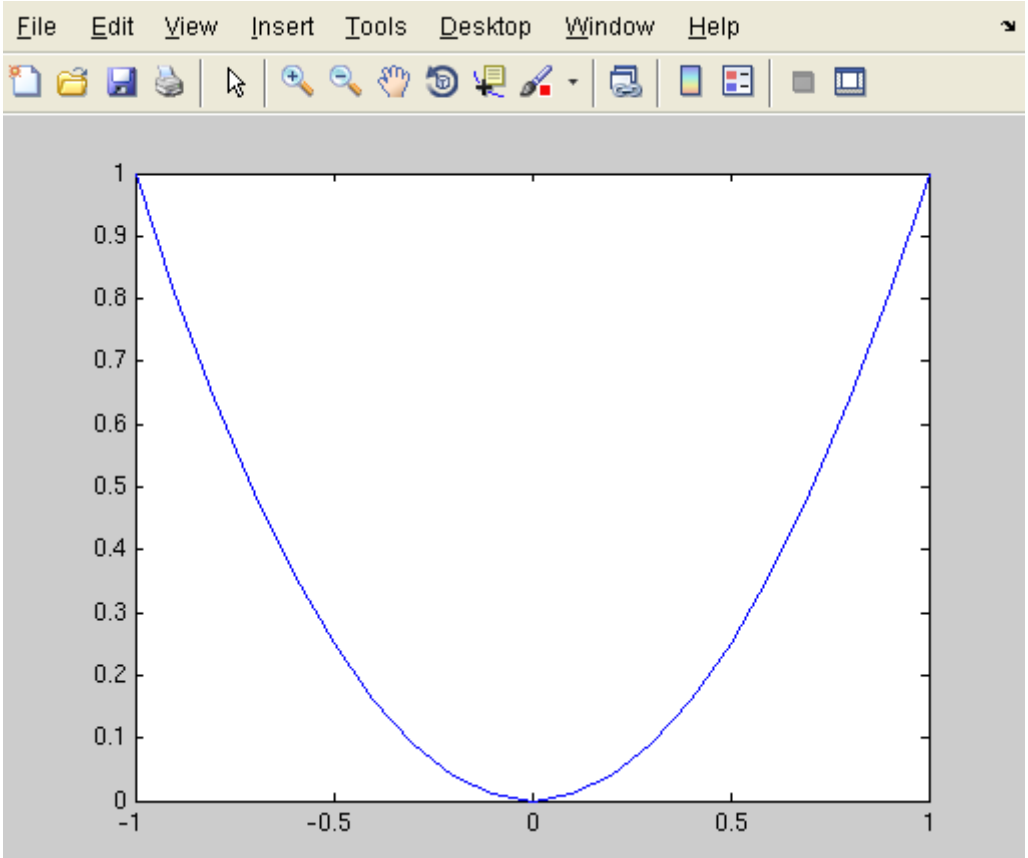

図 2：放物線をプロットした様子

5.3 複数のプロットを表示する方法

今のプロットに別の放物線を重ね書きするには、以下のように hold 命令を使います。

```
>> x = [-1:0.1:1];
>> y = x.^2;
>> hold off
>> plot(x, y)
>> hold on
>> plot(x, 2*y)
>> hold off
```

これにより、２つの放物線が重ね書きされる様子がわかります。ここで、2 乗を実現するために「.^」が使われていることに注意してください（ピリオドを付けることによって、演算が要素ごとに適用されます)。このように、hold 命令は on にしたり off にしたりすることで、重ね書きを許可したり許可しなかったりを制御します。通常は許可されていませんので、plot 命令を実行するたびにプロット結果は上書きされていきます。

ところでこの例では同じ線を用いているために区別がつきにくいことがあります。その場合、線の色や種類を変えることもできます。2 本目のプロットを以下のように変えてみてください。

```
>> plot(x, 2*y, 'r')
```

これによって、2 本目が赤になります。また、

```
>> plot(x, 2*y, 'r:')
```

とすることによって 2 本目がさらに点線になります。

このように、同一のプロット画面上に重ね書きする方法はわかりました。それでは、2 つのプロットを並べることを次に考えます。その際、用いるのが subplot になります。

```
>> x = [-1:0.1:1];
>> y = x.^2;
>> subplot(2,1,1)
>> plot(x, y)
>> subplot(2,1,2)
>> plot(x, 2*y)
```

これによって、２つのプロットが縦に並びます。横に並べるには、

```
>> subplot(1,2,1)
>> plot(x, y)
>> subplot(1,2,2)
>> plot(x, 2*y)
```

このように、subplot の 1 番目の数字（引数）はプロット画面の行数を表し、2 番目の引数は列の数を表します。そして 3 番目に通し番号を書くことによって、その直後で実行されるプロット命令が何番目に描画されるかが決まります。

以上、2 つの方法を紹介しました。しかし、いずれもが同一ウィンドウ内で描画されていました。違うウィンドウに別々にプロットすることも可能です。そのためには figure 命令を用います。

```
>> x = [-1:0.1:1];
>> y = x.^2;
>> figure(1)
>> plot(x, y)
>> figure(2)
>> plot(x, 2*y)
```

これによって、新しいウィンドウに 2 つ目のプロットがされるのがわかります。

5.4 図を整える

ただ単に plot 関数を使って表示した図にはタイトルや軸の名前がなかったり、表示範囲や目盛りが思うようになっていなかったりします。そこで、図をきれいに「整える」必要があります。それには、図 3 にあるようにプロット画面のメニューバーにある「Edit」を選びます。その中のプルダウンメニューの中に「Axes Properties」というのがありますので、

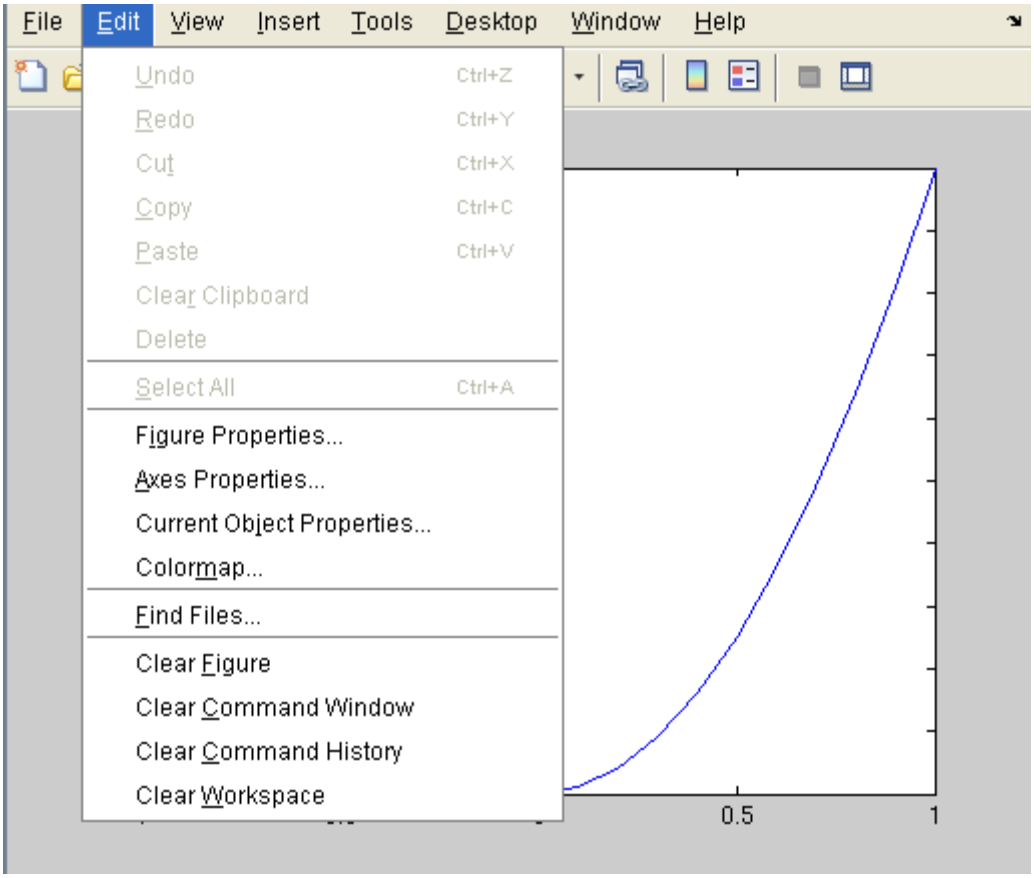

図 3：プロットの画面でメニューバーから「Edit」を選んだ様子

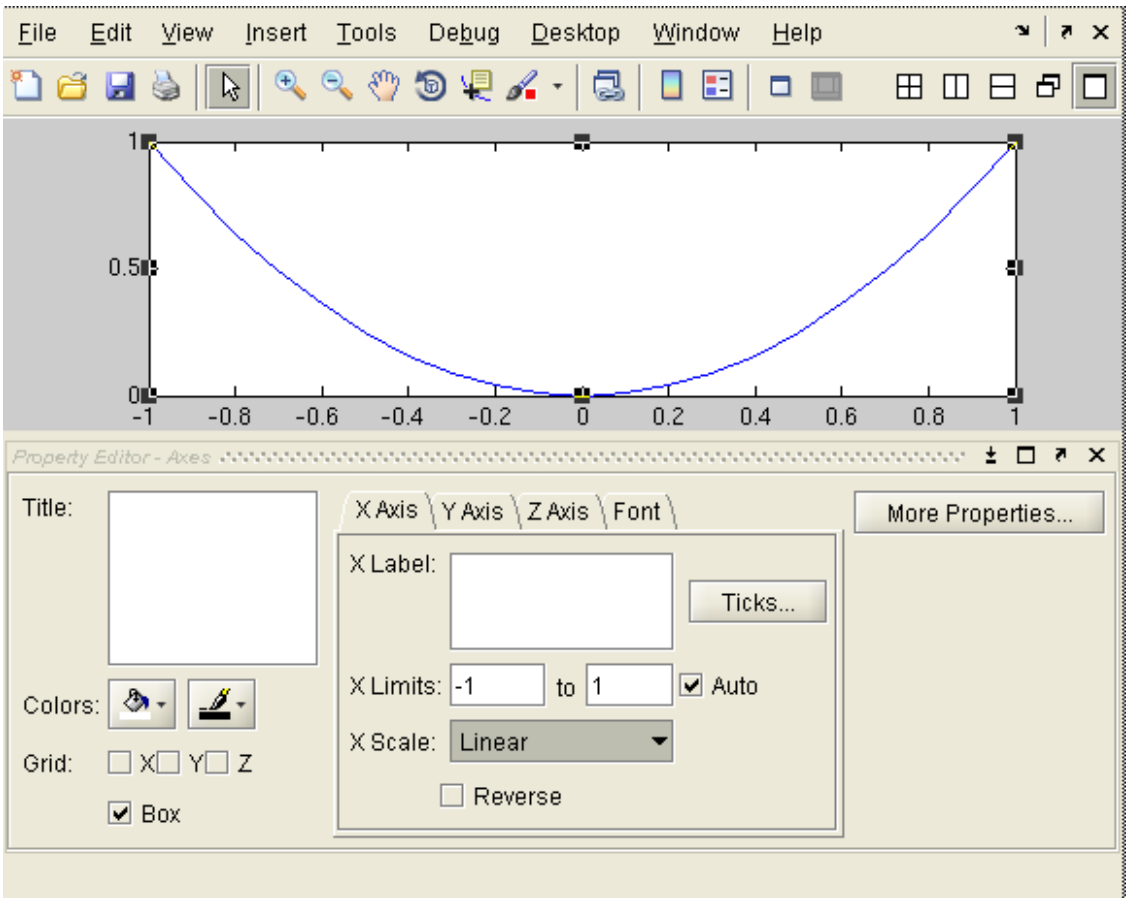

図 4：「Axes Properties」を選んだ後に登場するウィンドウの様子

それを選ぶと図 4 のようなウィンドウが出てきます。このウィンドウの中で様々な設定を行うことが可能となります。例えば、X-軸を操作するには「X-Axis」を選択します。X-軸の範囲を決めるのが「X Limits」です。「Auto」が選択されている場合には、自動的に範囲が決まります。目盛りのマーカに関わるのが「Ticks」です。軸に名前を書く場合には「X Label」を使います。目盛りを対数尺度にするには、「X Scale」を「Log」にしてください。「Linear」が選択されている場合、目盛りは線形尺度となります。グリッドを出力する場合は、「Grid」を選択します。

このように軸に関してのプロパティを変えられる他、図全体のタイトルを書くことも可能です。また、プロットされているグラフの線の種類や太さなど、各オブジェクトのプロパティも同様にいろいろと設定・編集することが可能です。

5.5 ３次元プロット

３次元プロットを実現するにいくつかの関数が用意されていますが、その代表的なものが

mesh や surf といったものです。例えば

```
>> mesh(A)
```

を実行することによって、行列の中身を３次元データとして描画することができます。また対象を回転させるなどして、見易い角度に視点を変えることも可能です。

6. その他

6.1 行列を作る

予め、空の行列を作っておきたいことがしばしばあります。そのような場合には、

```
>> A = zeros(4,5);
```

などを実行します。この場合、A4 行 5 列の行列となり、その各要素は 0 となります。

これに似たものに、ones というのがあります。

```
>> A = ones(4,5);
```

これを実行すると、先ほどと同様に A は 4 行 5 列の行列になりますが、各要素は 1 となります。

各要素を乱数で埋めたい場合には、

```
>> A = rand(4,5);
```

を実行します。この場合、乱数は一様乱数となりますが、正規分布に基づく乱数を用いたい場合は、

```
>> A = randn(4,5);
```

とします。

6.2 データの読み込みと保存

例えば、行列の中身を保存した場合には save を用います。

```
>> save output x
```

などを実行することによって、変数 x を外部ファイルとして保存することができます。このとき、ファイル名は「output.mat」というように、.mat という拡張子がつきます。このファイルはバイナリファイルで、中身を Matlab 以外のアプリケーションで見ることは通常できません。もし他のアプリケーションで使いたい場合は、

```
>> save output x -ascii
```

というように「-ascii」というオプション指定をします。そうすると、外部ファイルはアスキー形式のファイルとなり、例えばエクセルでもデータを取り込むことが可能となります。

逆に、保存されているファイルからデータを読み込む場合には、「load」命令を用います。例えば、先ほど保存した「output.mat」からデータを読み込む場合は、

```
>> load output
```

を実行します。すると、変数 x が Matlab 上にロードされることになります。もしアスキー形式で保存されたファイルから読み込む場合は、

```
>> load output -ascii
```

というように「-ascii」オプションを用います。

6.3 M ファイル

Matlab での一連の記述をいちいちプロンプトが出る度に 1 行 1 行次々と手動で入力することはしばしば大変なことがあります。そこで、その一連の記述を別のファイルとして保存しておき、そのファイルを「foo.m」というように.m という拡張子をつけた名前にしておくと、そのファイルを Matlab の中から実行することができます。この場合、Matlab 上では

```
>> foo
```

と実行するだけで、そのファイルの中に実際に書かれているステートメントが実行され、いちいち手動で入力する手間が省けます。

6.4 Matlab 上で 1 行に長い記述をしたくない場合

Matlab において、行をまたがるような記述をせざるを得ない場合があります。しかしそのような場合は煩雑になり、みにくくなることが往々にしてあります。そのような場合、ピリオドを 3 つ連続させた「...」を行末に書くことによって、その行と次の行が 1 つの行として見なされます。例えば、

```
>> x = [10 11 12 13 14 15 16 17 18 19 ...
       20 21 22 23 24 25 26 27 28 29];
```

などと使います。

6.5 Matlab 上で wav ファイルを扱う

Matlab で作成されたデータを MS-Windows 用の wav ファイル（音のためのファイル）として入出力するには、wavread や wavwrite という関数を用います。

もし、列ベクトル x の中身を wav ファイルとして保存する場合には、

```
>> wavwrite(x, 44100, 'output.wav')
```

とします。ここで、44100 はサンプリング周波数を現します。この場合、x はベクトルデータ、つまり 1 次元データであるとしましたが、もし x が 2 列からなる行列である場合、保存される wav ファイルはステレオファイルとなります。

[注意]
wavwrite を実行する前に、x の中身を±1 の範囲に抑えておくことが必要になります。もしその範囲を越えている状態で wavwrite を実行すると、雑音を産む原因となるので注意が必要です。

一方、すでに存在する wav ファイルを Matlab に取り込むには、例えば

```
>> y = wavread('output.wav');
```

などとします。この場合、もし output.wav がモノラルであれば y の列の数は 1 に、ステレオであれば 2 になります。また、サンプリング周波数の情報も同時に取得したい場合は、

```
>> [y Fs] = wavread('output.wav');
```

とします。これによって、Fs の中にサンプリング周波数が取り込まれます。

7. おわりに

Matlab の基本的な使い方について説明して参りましたが､Matlab ではもっといろいろなことが可能です。様々な参考書が出ていると同時に、オンラインのヘルプも充実していますので、それらを参考にしながら多くの機能を学ぶことによってさらなる世界を体験し活用していっていただきたいと思います。

謝辞

このテキストを作るにあたり、上智大学理工学部情報理工学科荒井研究室のメンバー、特に安啓一さん、篠田貴彦さんにお世話になりました。ありがとうございました。

Matlab の使い方

初版　２００２年４月　発行(２００)
2002.12(200)
2004.5 (200)
2005.9 (200)
第２版　２００９年４月　発行(２００)
2010.1 (200)
2011.1 (200)
第３版　２０１４年１２月　発行

著者　荒井隆行
発行　上智大学総合メディアセンター
住所　〒102-8554　千代田区紀尾井町７－１
（電話）03-3238-3101（直通）