MATLAB TUTORIAL

This is not a matlab tutorial by any stretch of imagination. This is just a short introduction to using matlab to for matrix multiplication and inversion, plotting, and for complex number manipulation. Jut enough to what you need in a basic circuit course.

Introduction

Matlab is a very powerful tool for numerical calculation that is extensively used by engineers. The syntax of Matlab is very close to C syntax which makes it easy for anyone with a background in C or Java.

- In matlab every variable is an array (matrix), even a scalar is considered as 1x1 matrix. You have to be very careful when multiplying or adding since the matrix dimension should be compatible.
- You do not need to declare variables before you use them. Variables are declared on the fly when you use them.
- If you type a=5 that creates a variable a (1x1 matrix) and assign 5 to it. If you type a=[1 2 3] although a was defined to be 5, the old variable will be gone and a new variable a is created as 1x3 matrix with value [1 2 3].
- If you end the statement with a semicolon, the calculated value will not be echoed to the screen, If you do not, the calculated (or defined) value will be echoed to the screen.
- If you type a mathematical expression without assigning its value to a variable (like 3+5 or a*b) the value will be assigned to a variable called "ans"
- Comments start from % to the end of the line.

The best thing to do now is to start Matlab and go through a simple exercise. Start matlab from the start windows on the computer, you will get a screen like this (Note that the starting screen for you is different since you can control what windows will be shown, and the history is shown and mine is definitely different than you).
The screen shot is too small; I will go through the commands one by one for you. first I typed at the >> prompt. My commands are shown in red, while Matlab response is shown in blue

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

Note there is no semicolon after that, the variable is created, initialized and echoed to the screen.
Then I typed a=a+1;

Since a is a 3x1 matrix, and 1 is a scalar (1x1), 1 is expanded to [1 1 1] and added to a (add 1 to every element in a). The operation is completed and nothing is displayed since I added “;” after the statement. However no a is [2 3 4]

```
>> a=a+1;
```

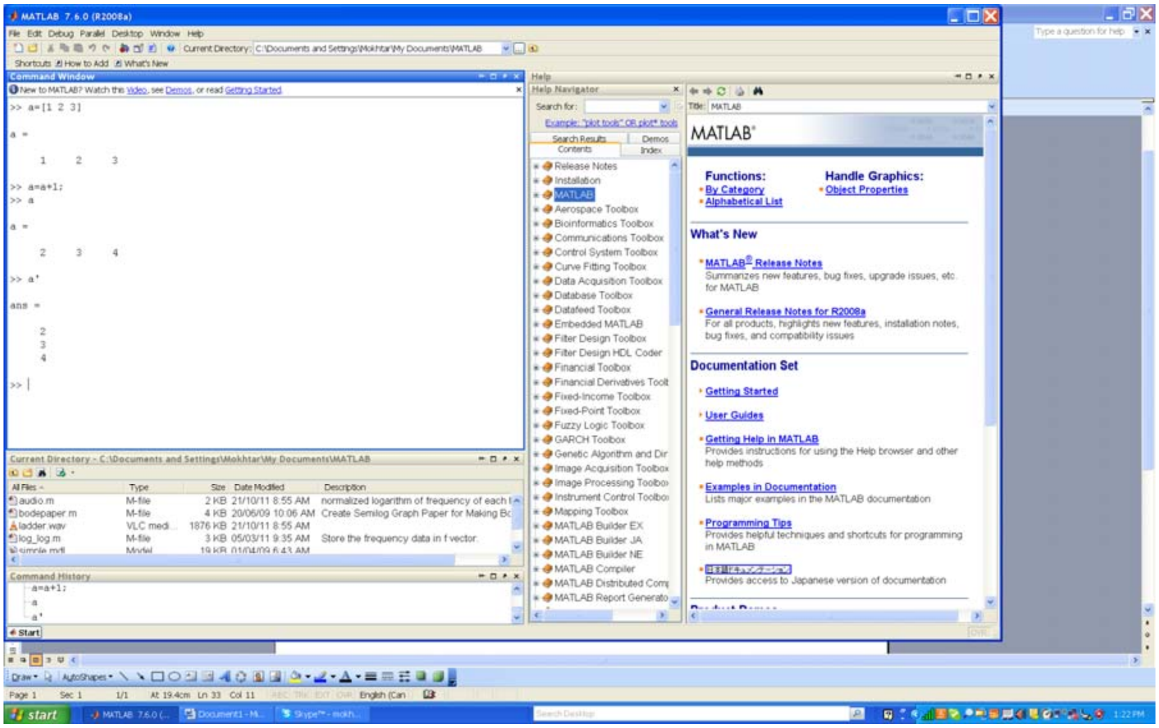
next I typed a (no semicolon) Matlab echoes the value of a

```
>> a

a =

     2     3     4
```

then I typed a’ (that is the transpose of a, a is transposed and echoed to the screen as shown below

```
>> a'

ans =

     2
     3
     4


>>
```

Not that the value of a is still [2 3 4] did not change.



MATLAB 7.6.0 (R2008a)
File Edit Debug Parallel Desktop Window Help
Current Directory: C:\Documents and Settings\Mokhtar\My Documents\MATLAB
Shortcuts How to Add What's New
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

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

a =

     1     2     3

>> a=a+1;
>> a

a =

     2     3     4

>> a'

ans =

     2
     3
     4

>> |
```

Current Directory - C:\Documents and Settings\Mokhtar\My Documents\MATLAB

All Files	Type	Size	Date Modified	Description
audio.m	M-file	2 KB	21/10/11 8:55 AM	normalized logarithm of frequency of each t
bodepaper.m	M-file	4 KB	20/06/09 10:06 AM	Create Semilog Graph Paper for Making Bc
ladder.wav	VLC medi...	1876 KB	21/10/11 8:55 AM	
log_log.m	M-file	3 KB	05/03/11 9:35 AM	Store the frequency data in f vector.
simple.mdl	Model	19 KB	01/04/09 6:43 AM	

Command History

```
a=a+1;
a
a'
```

MATLAB® Release Notes
Summarizes new features, bug fixes, upgrade issues, etc for MATLAB

General Release Notes for R2008a
For all products, highlights new features, installation notes, bug fixes, and compatibility issues

Getting Help in MATLAB
Provides instructions for using the Help browser and other help methods

Examples in Documentation
Lists major examples in the MATLAB documentation

Programming Tips
Provides helpful techniques and shortcuts for programming in MATLAB

日本語ドキュメンテーション
Provides access to Japanese version of documentation

M files

Entering the commands one by one might not be the easiest way to do this. You can put all your commands in a file and run this file as a batch file. To do this Choose the Menu file ->new->M-File
A new editor window is shown, write in it the copmmands that you want. Type the following

```
a=12;
b=3;
c=sqrt(a*b)+11
```

Save this file as test1.m
The type test1 at the >> prompt (you might get a window saying this file is not in the path, choose added to the path. From now on Matlab will search in the directory test1 is in to look for m-files).
matlab output will look as

```
>> test1

c =

   17

>>
```

The output is the same as if you typed the statements in the file interactively into Matlab window.
From now on, you may enter the statements (commands) interactively, or write them in an M-file and execute the file.

Basic Matrix Operations

The basic operations are + - * / and ^ (^ is raised to the power of).
All these operations are matrix operation, so the matrix dimensions should be compatible.
If we say c=a x b then the number of columns of a should be the smae as the number of rows of b.
However, Matlab allows an element by element operations. The element by element operations is done by adding a period (.) before the operator. So, * is multiplication, but .* is an element by element multiplication. Consider the following short program

Note that the answer is the element by element multiplication of a and b

```
>> a=[1 2 3];
>> b=[4 5 6];
>> a*b
??? Error using ==> mtimes
Inner matrix dimensions must agree.

>> a.*b

ans =

     4    10    18
```

Consider the following system of equations.

$$\begin{bmatrix} 1 & 1 & 3 \\ 2 & 4 & 1 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 8 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 3 \\ 2 & 4 & 1 \\ 3 & 2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ 6 \\ 8 \end{bmatrix}$$

Here is a program to solve this in Matlab (done interactively)

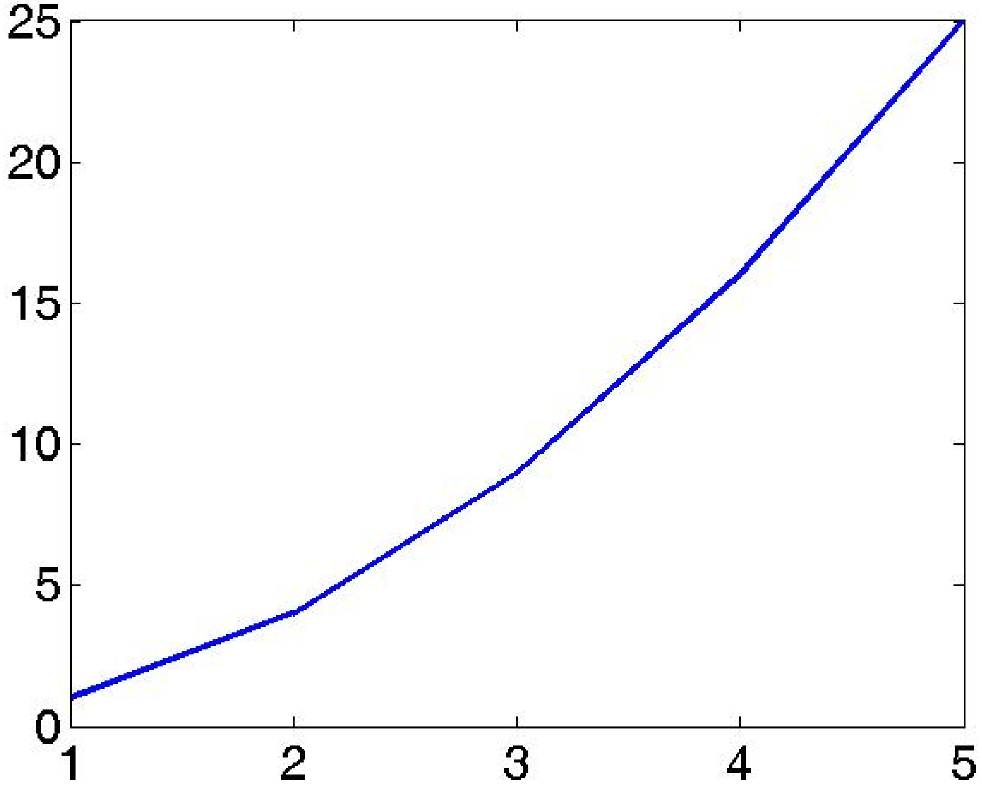
```
>> a=[1 1 3; 2 4 1; 3 2 1];
>> b=[3 6 8]'

b =

     3
     6
     8

>> c=inv(a)*b

c =

    2.4762
    0.2381
    0.0952

>>
```

Plotting

The function plot(x,y) plots two vectors. Both x,y should be of the same dimensions. Consider the following small script

```
>> x=[1 2 3 4 5];
>> y=x.^2;
>> plot(x,y)
>>
```

Note that x is defined as [1 2 3 4 5], they y=.^2 that is an element by element (the dot before ^) operation. Then y=[1 4 9 16 25], the following graph is displayed which you can save as jpg or png (use File->save as).



That is the relation y=x^2.
To draw the equation y=1-e^{-5t} in the range 0<t<1
Then we define t as a vector that goes from 0 to 2. The number of points in the curve determine how smooth the curve will be. Consider the following
t=[0:0.1:2] that creats a vector [0 ,0.1, 0.2, 0.3, … 2].

Assumethat we want to plot 2 functions in the same graph, the first is the above exponential function, and a linear function y1=t
The following script does that

```
t=[0:0.1:2];
y=1-exp(-5*t);
x=t;
plot(t,y,'-x',t,x,'--');
legend('exp', 'linear')
xlabel('time');
ylabel('y value');
```

the first line defines t as from 0 to 5 with a step of o.e

The second line defines y as the above exponential function

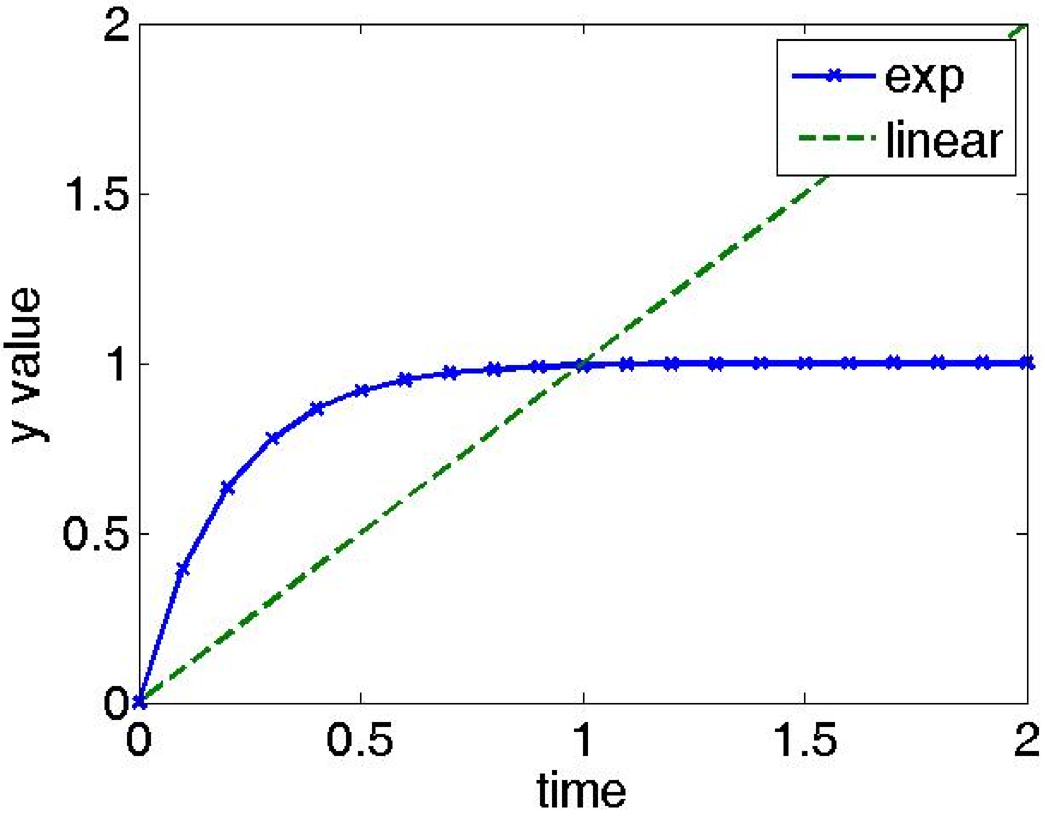
The third line defines a function x=t (linear)

The fourth line plots t,y and t,x on the same graph. t,y followed by ‘-x’ meaning the line is solid with x at every point. t,x followed by ‘--‘ meaning the line is dashed

The fourth line gives a legend for these 2 curves.

The fifth and sixth line put the caption for the x and y axis

The resulting graph is shown below



Check help plot for moe details.

Complex numbers

Matlab can represent complex numbers. it takes *i* or *j* as the square root of 1. For example the number 1+2j is represented as 1+j*2
Consider the following script, it is self explanatory.

```
>> a=1+i*3

a =

   1.0000 + 3.0000i

>> b=2+j*4

b =

   2.0000 + 4.0000i

>> c=a*b

c =

 -10.0000 +10.0000i

>> abs(c)

ans =

   14.1421

>> real(c)

ans =

   -10

>> imag(c)

ans =

    10

>>
```

where the multiplication is complex multiplication, real is the real part of the number, imag is the imaginary part, and abs is the absolute value or magnitude of the number