

ELG3125

# Signal and System Analysis Lab

- **Lab1: Introduction to MATLAB**

By: Hitham Jleed

[hjleed@uottawa.ca](mailto:hjleed@uottawa.ca)

# Getting Started

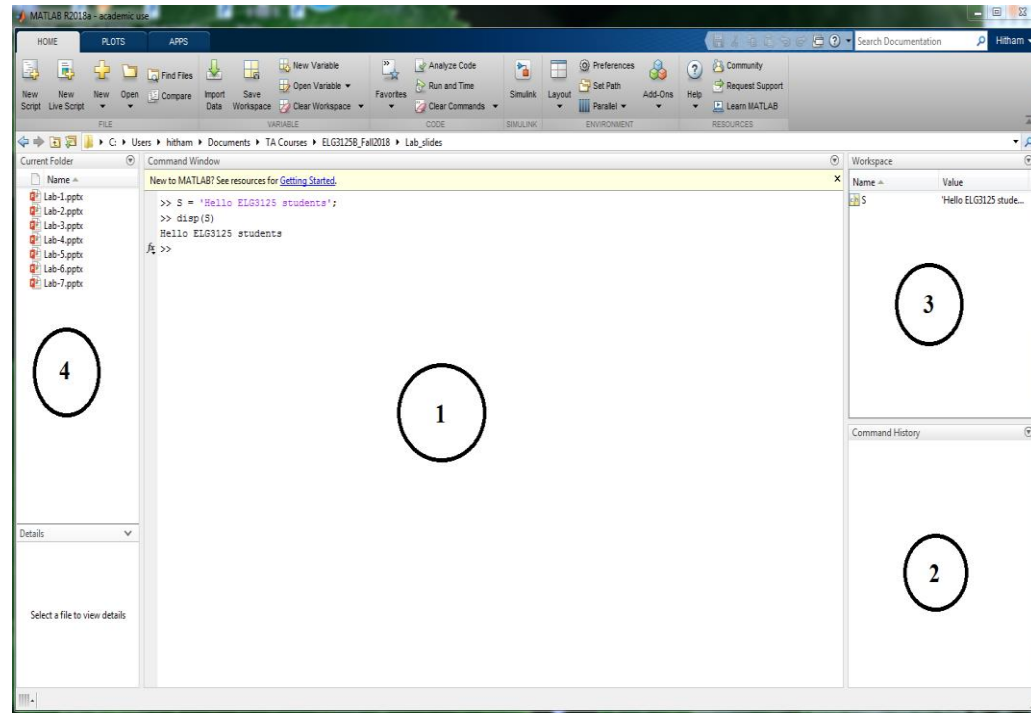
What is MATLAB?

MATLAB is produced by MathWorks, and is one of a number of commercially available software packages for numerical computing and programming. MATLAB provides an interactive environment for algorithm development, data visualisation, data analysis, and numerical computation.

**MATLAB** derives its name from **MAT**rix **LAB**oratory.

# the MATLAB environment

1. **Command Window:** This is the main window, and contains the command prompt (`>>`). This is where you will type all commands.
2. **Command History:** Displays a list of previously typed commands. The command history persists across multiple sessions and commands can be dragged into the Command Window and edited, or double-clicked to run them again.
3. **Workspace:** Lists all the variables you have generated in the current session. It shows the type and size of variables, and can be used to quickly plot, or inspect the values of variables.
4. **Current Directory:** Shows the files and folders in the current directory. The path to the current directory is listed near the top of the MATLAB desktop. By default, a MATLAB folder is created in your home directory on your M:drive, and this is where you should save your work.



# Basic Calculations

MATLAB can perform basic calculations such as those you are used to doing on your calculator. E.g.:

## Addition

```
>> 5+5
ans =
    10
```

## Exponentiation

```
>> 2^3
ans =
     8
```

## Trigonometry

```
>> sin(2*pi)+exp(-3/2)
ans =
    0.2231
```

## Complex numbers

```
>> 5+4j
ans =
    5.0000 + 4.0000i
```

### Comments:

- MATLAB has pre-defined constants e. g.  $\pi$  may be typed as pi.
- Complex numbers can be entered using the basic imaginary unit i or j.

### (?) Built-in functions

There are many other built-in MATLAB functions for performing basic calculations. These can be searched from the Help Browser which is opened by clicking on its icon (like the icon used to indicate this Hints and Tips section) in the MATLAB desktop toolbar.

### Arithmetic operations command description

|   |                     |
|---|---------------------|
| + | Addition            |
| - | Subtraction         |
| * | Multiplication      |
| / | Division            |
| ^ | Exponentiation      |
| ' | Conjugate transpose |



# Basic Math Functions

- $\text{abs}(x)$  absolute value
- $\text{exp}(x)$  exponential
- $\text{sin}(x), \text{cos}(x)$  sine, cosine
- $\text{log}(x), \text{log}_{10}(x)$  natural logarithm, common logarithm
- $\text{sqrt}(x)$  square root
- $\text{sign}(x)$  signum
- $\text{round}(x), \text{fix}(x)$  round towards nearest integer, round towards zero
- $\text{floor}(x), \text{ceil}(x)$  round towards negative infinity, round towards plus infinity
- $\text{size}(x), \text{length}(x)$  size of array, length of vector



# Variables and Arrays

- A variable is a symbolic name associated with a value. The current value of the variable is the data actually stored in the variable.
- Variable name:
  - up to 63 characters (as of MATLAB 6.5 and newer).
  - Case sensitive. e.g., x and X are two different variables.
  - must start with a letter and can be followed by letters, digits, or underscores. e.g., **x3\_2** is correct, but **2\_x3** is not correct.
- Variables are stored in MATLAB in the form of matrices which are generally of size  $M \times N$ .
- Elements of matrix can be real or complex numbers.
- A scalar is a  $1 \times 1$  matrix.
- A row vector is a  $1 \times N$  matrix.
- A column vector is a  $M \times 1$  matrix.

# MATLAB Symbols

- `>>` Command prompt
- `...` Continue statement in next line
- `,` Separate statements and data, e.g. `A = [5.92, 8.13, 3.53]`
- `%` Start comment which ends at the end of line, e.g.,  
`% Here you can type what you want`
- `;` Suppress output or used as row separator in a matrix
- `:` Specify a range and generates a sequence of numbers that you can use in creating or indexing into arrays. For example, `N = 1:10`.

## Basic Examples

### Using the whos command

```
>> a = 2
a =
    2
>> b = 3
b =
    3
>> edinburgh = a+5
edinburgh =
    7
>> whos
```

| Name      | Size | Bytes | Class  | Attributes |
|-----------|------|-------|--------|------------|
| a         | 1x1  | 8     | double |            |
| b         | 1x1  | 8     | double |            |
| edinburgh | 1x1  | 8     | double |            |

```
>> x = [1 2 3] % Creating a row vector
x =
    1    2    3
>> y = [4; 5; 6] % Creating a column vector
y =
    4
    5
    6
>> y' % The transpose operator
ans =
    4    5    6
>> z=3:2:9 % Creating vectors using ranges
z =
    3    5    7    9
>> v=linspace(0,10,5)
v =
    0    2.5000    5.0000    7.5000   10.0000
```

### (?) *clear and clc commands*

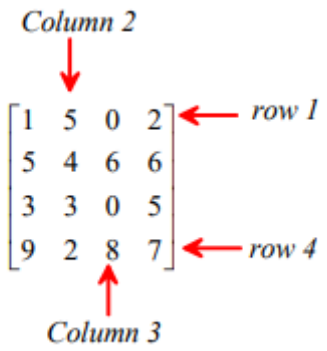
The **clear** command can be used if you want to clear the current workspace of all variables. Additionally, the **clc** command can be used to clear the Command Window, i.e. remove all text.



# Matrix Operation

MATLAB excels at matrix operations, and consequently the arithmetic operators such as multiplication (\*), division (/), and exponentiation (^) perform, by default, matrix operation, when used on a vector. To perform an element-by-element multiplication, division, or exponentiation you must precede the operator with a **dot**.

```
>> a=[1 5 0 2;5 4 6 6;3 3 0 5;9 2 8 7]
a =
    1    5    0    2
    5    4    6    6
    3    3    0    5
    9    2    8    7
```



```
>> a=[2 3;5 1]
a =
    2    3
    5    1
>> b=[4 7;9 6]
b =
    4    7
    9    6
```

```
>> a*b % Matrix multiplication
ans =
    35    32
    29    41
>> a.*b % dot product
ans =
    8    21
    45    6
```

We can select the k-th element of a vector using subscripted indexing. For example:

```
>> a = [-2,3,-1,5,7]
a =

    -2     3    -1     5     7
>> a(1)
ans =

    -2
>> a(2:3)
ans =

     3    -1
>> idx=2:4
idx =

     2     3     4
```

```
>> idx=2:4
idx =

     2     3     4
>> a(idx)
ans =

     3    -1     5
>> idx1=find(a>0)
idx1 =

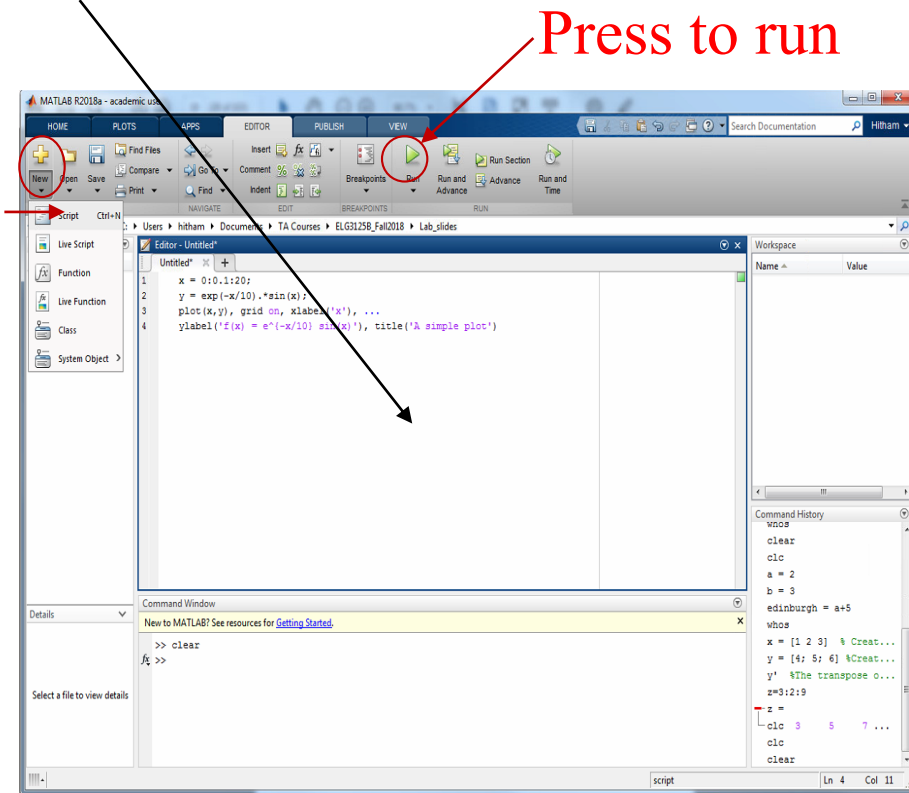
     2     4     5
>> a(idx1)
ans =

     3     5     7
```

# M-Files

MATLAB allows writing two kinds of program files –

- **Scripts** – script files are program files with **.m extension**. In these files, you write series of commands, which you want to execute together. Scripts do not accept inputs and do not return any outputs. They operate on data in the workspace.
- **Functions** – functions files are also program files with **.m extension**. Functions can accept inputs and return outputs. Internal variables are local to the function.



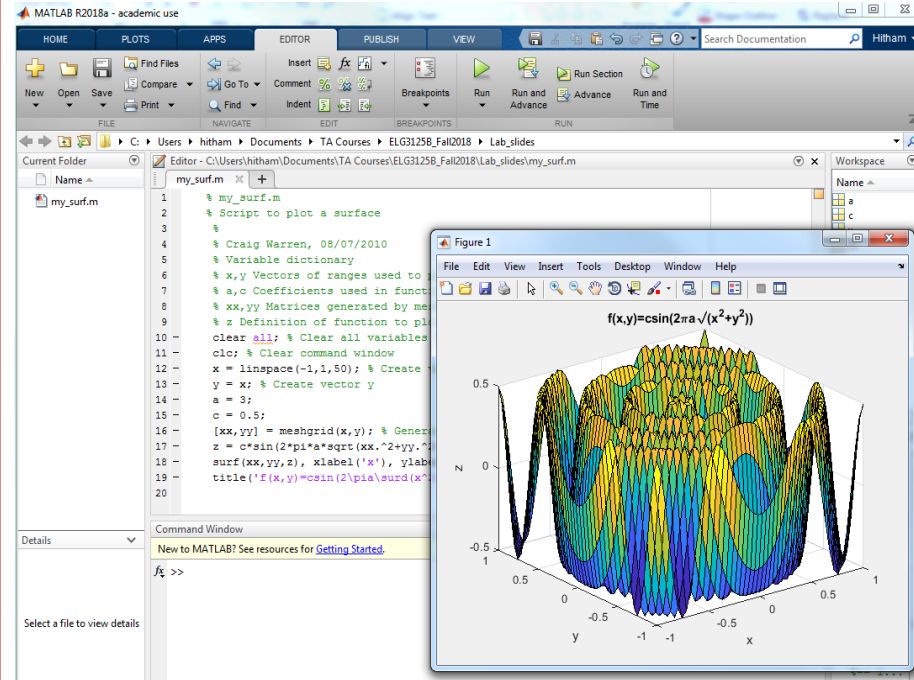
The script can be run by either typing its name at the command prompt, or clicking the *Save and run* icon

# Example of M-file

```

% my_surf.m
% Script to plot a surface
%
% Craig Warren, 08/07/2010
% Variable dictionary
% x,y Vectors of ranges used to plot function z
% a,c Coefficients used in function z
% xx,yy Matrices generated by meshgrid to define points on grid
% z Definition of function to plot
clear all; % Clear all variables from workspace
clc; % Clear command window
x = linspace(-1,1,50); % Create vector x
y = x; % Create vector y
a = 3;
c = 0.5;
[xx,yy] = meshgrid(x,y); % Generate xx & yy arrays for plotting
z = c*sin(2*pi*a*sqrt(xx.^2+yy.^2)); % Calculate z (function to plot)
surf(xx,yy,z), xlabel('x'), ylabel('y'), zlabel('z'), ...
title('f(x,y)=csin(2\pi*a\surd(x^2+y^2))') % Plots filled-in surface

```



## Comments:

- It is extremely useful, for both yourself and others, to put comments in your script files. A comment is always preceded with a percent sign (%) which tells MATLAB not to execute the rest of the line as a command.
- Script file names MUST NOT contain spaces (replace a space with the underscore), start with a number, be names of built-in functions, or be variable names.
- It is a good idea to use the clear all and clc commands as the first commands in your script to clear any existing variables from the MATLAB workspace and clear up the Command Window before you begin.

# Compare a script and a function

(a) Write a script: In the main menu of Matlab, select

file -> new -> M-file

A new window will pop up.

Input the following comma

`x = 1:5;`

`y = 6:10;`

`g = x+y;`

and then save the file as `myscript.m` under your path `matlab/ELG3125`:

(b) Write a function: Create a new m le following the procedure.

Type in the commands:

`function g = myfunction(x,y)`

`g = x + y;`

and then save it as `myfunction.m`

(c) Compare their usage

run the commands one by one:

`>> myscript`

`>> g`

`g =`

7 9 11 13 15

Run command `clear` to remove all variables from memory

Run the commands one by one:

`>> x = 1:5;`

`>> y = 6:10;`

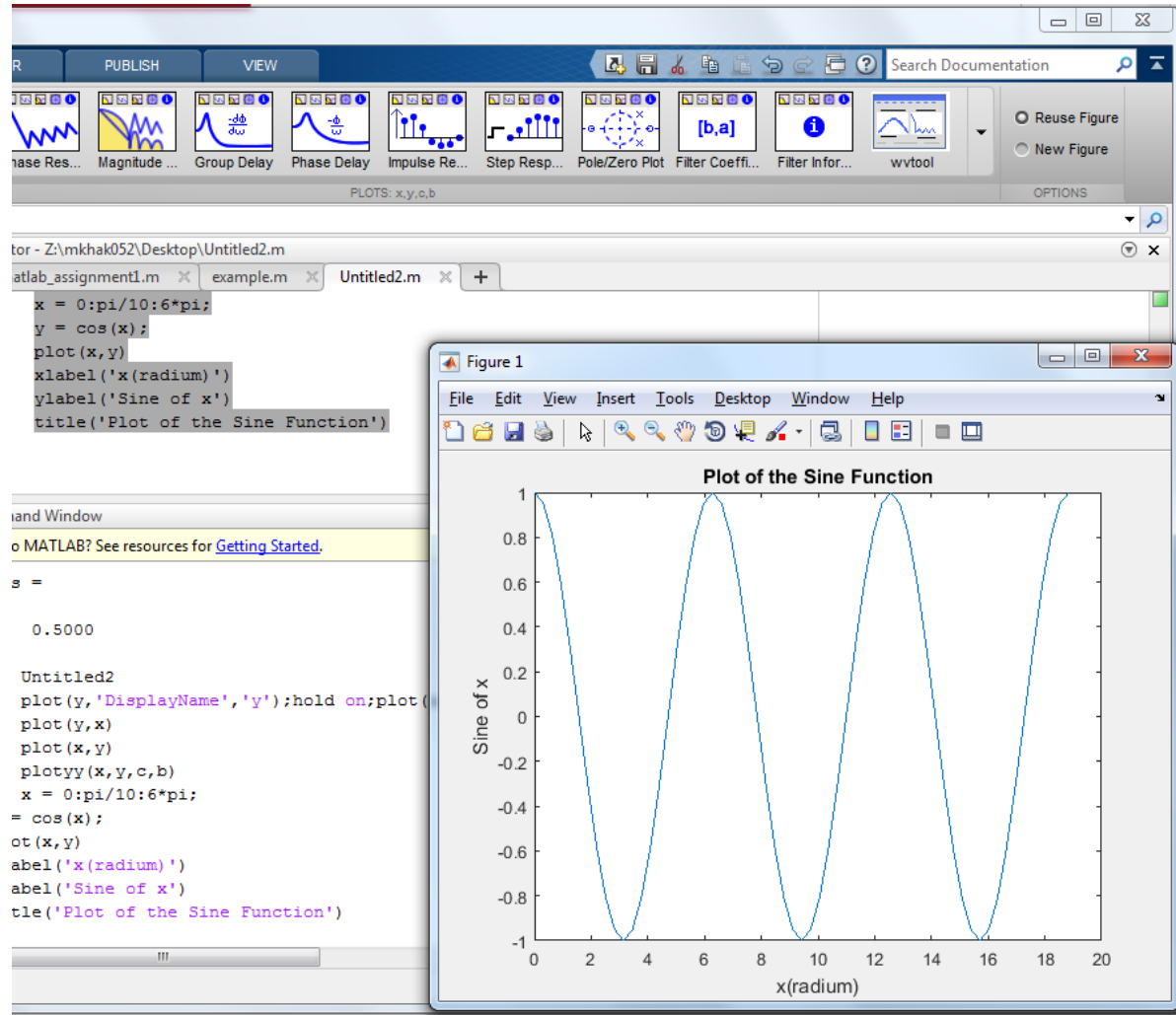
`>>z = myfunction(x,y)`

`z =`

7 9 11 13 15

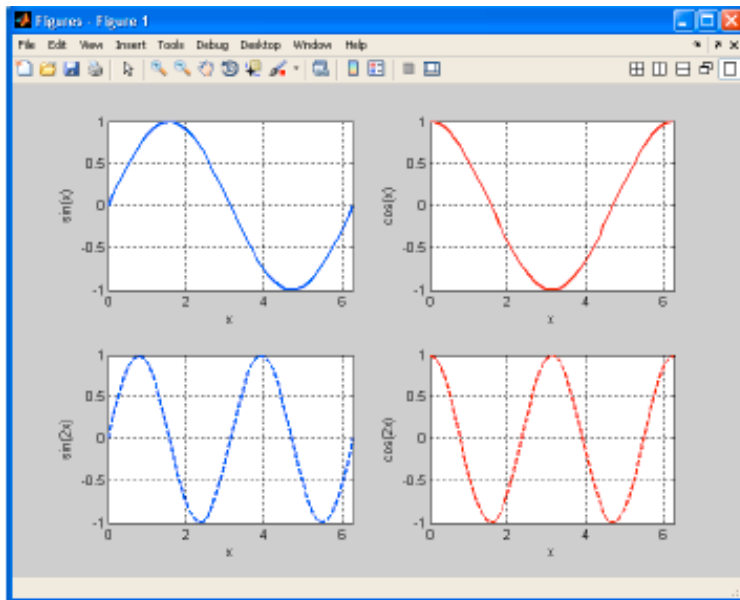
# Plot

```
x = 0:pi/10:6*pi;
y = cos(x);
plot(x,y)
xlabel('x(radium)')
ylabel('Sine of x')
title('Plot of the Sine Function')
```



# How to use subplot

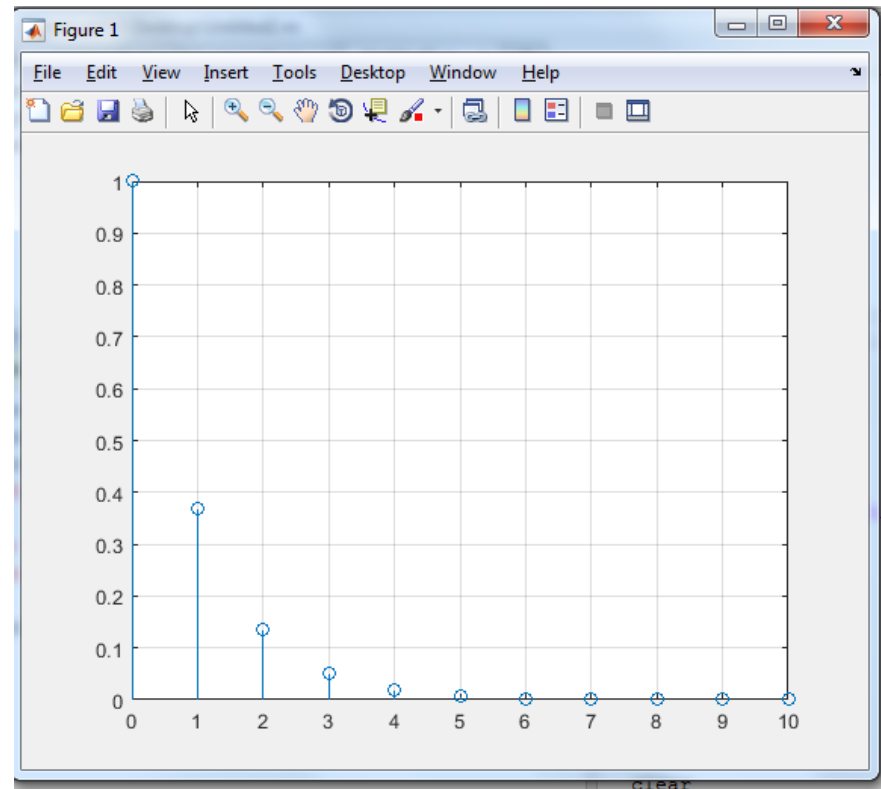
```
>> x = linspace(0,2*pi,50);  
>> subplot(2,2,1), plot(x,sin(x)), xlabel('x'), ylabel('sin(x));  
>> subplot(2,2,2), plot(x,cos(x)), xlabel('x'), ylabel('cos(x));  
>> subplot(2,2,3), plot(x,sin(2*x)), xlabel('x'), ylabel('sin(2x)');  
>> subplot(2,2,4), plot(x,cos(2*x)), xlabel('x'), ylabel('cos(2x)');
```



# How to use Stem

Let's show  $y[n] = e^{-n}$ :

```
n=0:10;  
y=exp(-n);  
stem(n,y),grid;
```





# Miscellaneous commands

Annotating your plots is essential to make them understandable (otherwise you quickly forgot what is what in the plot). Adding a title, x- and y-axis labels is easy to do using the commands `title`, `xlabel` and `ylabel` respectively. Adding a legend with names is also easy using the command `legend`. We can also manipulate the shape of the axis by using the `axis` command using a variety of arguments (see `help axis` for more). For better readability we can also make visible a grid using the `grid on` command.

```
>> help axis
axis Control axis scaling and appearance.
axis([XMIN XMAX YMIN YMAX])
sets scaling for the x- and y-axes
.....
.....
```

You can also use icon help from MATLAB desktop toolbar

```
>> t = linspace(0,2,500);
>> v = exp(-3*t);
>> plot(t,v)
>> grid on
>> title('Exponential decay')
>> xlabel('Time (sec)')
>> ylabel('Voltage (mV)')
>> axis square
```

Lastly two other useful commands are the `figure` command and the `close` command. With `figure` we tell MATLAB to create a new figure so that we don't overwrite

# Some plotting commands

|          |   |
|----------|---|
| plot     | Plot in linear coordinates as a continuous function |
| stem     | Plot in linear coordinates as discrete samples      |
| Loglog   | Logarithmic $x$ and $y$ axes                        |
| Semilogx | Linear $y$ and logarithmic $x$ axes                 |
| semilogy | Linear $x$ and logarithmic $y$ axes                 |
| bar      | Bar graph   |
| hist     | Histogram   |
| polar    | Polar coordinates                                   |

The simple 2D plotting commands

|                    |  |
|--------------------|--|
| xlabel             | Labels $x$ -axis   |
| ylabel             | Labels $y$ -axis   |
| title              | Puts a title on the plot   |
| grid               | Adds a grid to the plot  |
| axis               | Allows changing the $x$ and $y$ axes                               |
| figure             | Create a figure for plotting                                       |
| hold on            | Allows multiple plots to be superimposed on the same axes          |
| hold off           | Release hold on current plot                                       |
| close( $n$ )       | Close figure number $n$  |
| subplot( $a,b,c$ ) | Create an $a \times b$ matrix of plots with $c$ the current figure |

Customization of plots



# Loops

## 1. For Loops

## 2. While loop

```
>> for j=1:4,
    j
end
j=
1
j=
2
j=
3
j=
4
>>
```

```
>> v=[1:3:10]
V=
1 4 7 10
>> for j=1:4,
    v(j)=j;
end
>> v
1 2 3 4
>>
```

For example, find the first integer  $n$  for which  $\text{factorial}(n)$  is a 100-digit number:

```
n=12;
n = 1; n
nFactorial = 1;
while nFactorial < 1e100
    n = n + 1;
    nFactorial = nFactorial * n;
end
```

# Linear Algebra

MATLAB is also very good at solving systems of linear equations. For example, consider the equations:

$$3x_1 + 4x_2 + 7x_3 = 6$$

$$5x_1 + 2x_2 - 9x_3 = 1$$

$$-x_1 + 13x_2 + 3x_3 = 8$$

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{A} = \begin{bmatrix} 3 & 4 & 7 \\ 5 & 2 & -9 \\ -1 & 13 & 3 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 6 \\ 1 \\ 8 \end{bmatrix}$$

This system of equations can be expressed in matrix form as above

To solve these in MATLAB, you would simply type

```
>> A = [3,4,7;5,2,-9;-1,13,3]
```

```
>> b = [6;1;8]
```

```
>> x = A\b
```

(note the forward-slash, not the back-slash or divide sign) You can check your answer by calculating

```
>> A*x
```

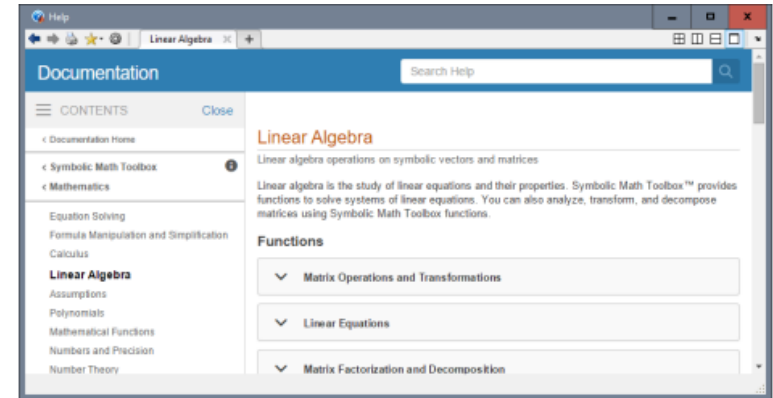
The notation here is supposed to tell you that x is b 'divided' by A – although 'division' by a matrix has to be interpreted rather carefully. Try also

```
>> x=transpose(b)/A
```

The notation  $\text{transpose}(\mathbf{b})/\mathbf{A}$  solves the equations  $\mathbf{x}\mathbf{A} = \mathbf{b}$ , where x and b are row vectors. Again, you can check this with

```
>> x*A
```

(The answer should equal b, (as a row vector) of course)



## Exercise

1. Write a Matlab program to solve and print

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} e & f \\ g & h \end{bmatrix} =$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} .* \begin{bmatrix} e & f \\ g & h \end{bmatrix} =$$

2. Write a MTLAB code which plots

$$y = \sin(4x) + \cos\left(x + \frac{\pi}{2}\right), 0 \leq t \leq 2\pi.$$

Enjoy MATLAB

End of Lab 1