HERIOT-WATT UNIVERSITY

DEPARTMENT OF COMPUTING AND ELECTRICAL ENGINEERING

B35SD2– Matlab tutorial 1
MATLAB BASICS

Objectives:

Matlab is a very powerful, high level language, It is also very easy to use. It comes with a wealth of libraries and toolboxes, that you can use directly, so that you don't need to program low level functions. It enables to display very easily results on graphs and images. To get started with it, you need to understand how to manipulate and represent data, how to find information about the available functions and how to create scripts and functions to generate programs.

During this session, you will learn:

1- How to start matlab.
2- How you can find out all the information you need.
3- How to create simple vectors and matrices.
4- What functions are available and how to find them.
5- How to plot graphs of functions.

After this hour, you will know most of what you need to know about Matlab and should definitely know how to go on learning about it on your own. So the "programming" aspect won't be an issue any more, and you will be able to use matlab as a tool to help you with you math, electronics, signal & image processing, statistics, neural networks, control and automation….Programming languages don't come any easier!

What is Matlab?

Matlab is a high performance language and software for technical computing. It allows

- Maths and computation
- Algorithm development
- Modelling, simulation and prototyping
- Data analysis and visualisation
- Scientific and engineering graphics

It is an interactive system (all data always in memory) with the following key facts:

- The basic data element is an array
- No declaration (or prototyping of functions) required
- All data represented as doubles
- In script mode or command line mode, data is always accessible and in memory.

Starting Matlab:

To start Matlab, double click on the matlab icon on the desktop. The main matlab window with the command line prompt ">>" should appear.

Matlab is an interpreted language. This means that the instructions you give to it are processed and interpreted by the computer directly, as opposed to for instance C: in C, when you write a program, you first need to compile it before it can be executed by the machine.
This means that at any moment, all the variables are in the memory of the computer (equivalent to debug mode of C/C++) and can be readily accessed. This is a tremendous help to quickly debug programs.

There is two ways to give instructions to matlab:

1- On the command line, after the prompt ">>": if you type an instruction, matlab executes it and gives you the result.
2- Through a script or a function, whose code is written in a text file with the .m extension. Scripts contain lists of instructions, just as could be written on the command line. To execute a script, just type the name of the .m file (without the .m extension) on the command line[1]. When a script is launched, all the instructions it contains are executed successively. Functions are modular scripts that can take in arguments. They won't be detailed here, but you can look at the on-line help to find out about them.

There are only a few data types in matlab. The main one is a matrix of doubles and most functions will only work with it. The second useful on is the uint (unsigned integer) type used to store images and limited to the [0,255] range.
Bear in mind that in matlab, all data is stored into matrices. A scalar (real) number is a 1 by 1 matrix. A vector is a 1 by N matrix. The command line gives you access to the so-called workspace. This is the collection of all the data and variables that you declare during a session.

Knowing where you are:

Use the pwd command to know where you are
Use the dir command to list the files in your current directory
Use the cd command to change directory
Now go to your home directory to be able to save today's work

The help command:

You can get help on all commands in matlab by typing (beware, matlab is case sensitive!):
>> help *command*

For instance, try typing
>> help help

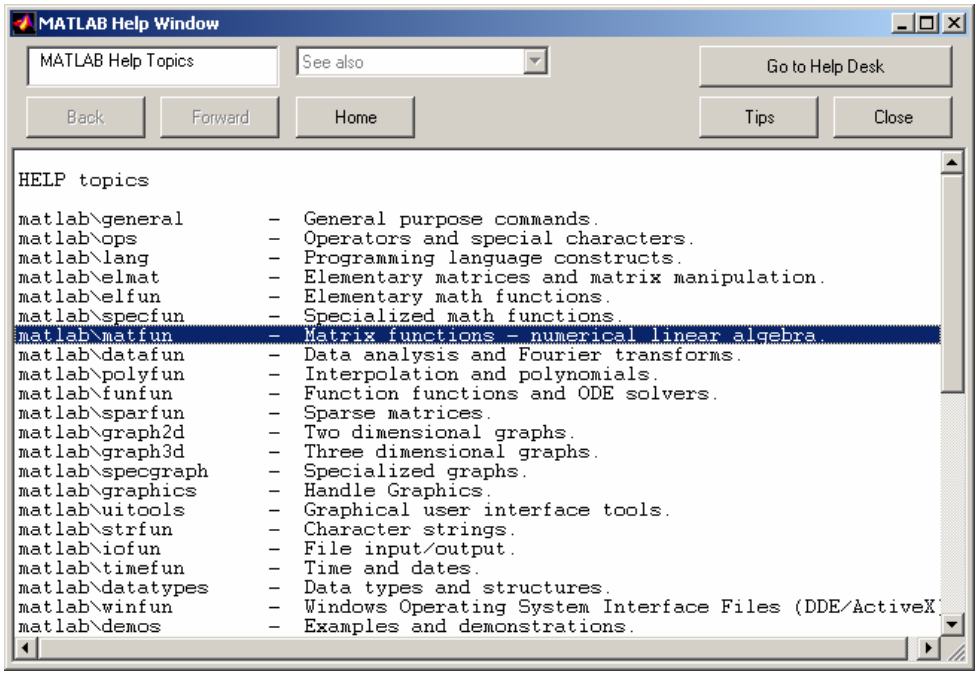
[1] Matlab needs to see this file. For this, it should either be physically in your current directory (type ">>pwd" to find out where you are), or the path to its location should be known by matlab (type ">>path" to see the full path; type ">>help path" if you need to change it). Go to the File menu and Set Path sub-menu to change the path.

And read what pops up in the screen…That should give you plenty of information on how to find information on topics. For instance, if you are looking for the cosine function, type

>> help cos

Note that if you press the "up" and "down" arrows on your keyboard, this enables you to recall previous commands.

You also have a more elaborated tool for help if the quick command line (usually faster) is not sufficient. To start it, press the ? button of the matlab window. This should start the window shown below.
For image processing functions, type help images to get the list of available functions and their short description.
For statistics, type help stats.



In this window, you will find all the main categories of matlab functions. Under each category, a full description of a specific function can be found.
Finally, the matlab manuals are available in pdf format at:

www.ece.eps.hw.ac.uk/~ceeyrp/ and follow the Teaching link to the image processing 22.5SD2 home page
or by clicking on the Go to help desk. If you press the getting started link, you will get a tutorial to get started with matlab. I recommend that you read it and try some of the functions and example. This help tool is very very useful.

The lookfor command:

If you do not know what is the name of the matlab command you want to use (which is usally the case!), the lookfor command is what you are looking for... very funny!

For instance, if you are looking for the cosine or related functions, type

>> lookfor cosine

matlab comes back with all matlab functions related to cosine with a short description as:

ACOS Inverse cosine.
ACOSH Inverse hyperbolic cosine.
COS Cosine.
COSH Hyperbolic cosine.
TFFUNC time and frequency domain versions of a cosine modulated Gaussian pulse.
DCT2 Compute 2-D discrete cosine transform.
DCTMTX Compute discrete cosine transform matrix.
DCTMTX2 Discrete cosine transform matrix.
IDCT2 Compute 2-D inverse discrete cosine transform.
CHIRP Swept-frequency cosine generator.
DCT Discrete cosine transform.
FIRRCOS Raised Cosine FIR Filter design.
IDCT Inverse discrete cosine transform.
DCT Discrete cosine transform.
IDCT Inverse discrete cosine transform.
dctold.m: %DCT Discrete cosine transform.
idctold.m: %IDCT Inverse discrete cosine transform.
BLKACOS This block defines an output angle that is the arccosine of the input.
BLKCOS This block defines the output as the cosine of the input.
BLKCOSASIN This block defines the cosine of an angle whose sine is u.
BLKCOSATAN This block defines the cosine of an angle whose tangent is u1/u2.

Now type help cos to get details on the cosine function.

>> help cos

Note that if you press the "up" and "down" arrows on your keyboard, this enables you to recall previous commands.

Creating a vector or a matrix:

As mentioned before, marices (and vector) as the basic element of a matlab program. matlab does see matrices as other languages see numbers and can perform operations in matrices directly without for loops, which comes handy for image processing...

Type :
>> x = [1 2 3 4 5 4 3 2 1];
>> x
>> who
>> whos
>> y = [6; 7; 8; 9; 0; 9; 8; 7; 6]
(note that is there is a semicolon at the end of the instruction, matlab doesn't expand the result of the instruction)
>>y'
>> z = [1 2 3; 4 5 6; 7 8 9; 0 1 2]

Operators:

MOST OF THEM WORK ON MATRICES DIRECTLY!

The colon operator:

A way to define quickly some vectors is to use the ":" operator. Type for instance (and observe the results):

```
>> a = 0:10
>> b = -5:10
>> c = 0:2:10
>> d = 10:-2:-5
>> e = 0:0.01:4.2
>> f = -pi:0.01:pi
```

You can see that these instructions are of the form *lower limit: increment: upper limit*

An other way to define elementary vectors that only contain ones and zeros: get help on *zeros*, *ones*, *randn* …
Try:

```
>> g = zeros(2,4)
>> h = ones(5,3)
```

Arithmetic operator:

These are the standard operators. (+ plus, - minus, / divide, * multiply, ^ exponent)

Try

```
>> 3^2
>> y'
>> x+y'
>> x*y
>> 3*x
```

The operators (/ , * and ^) have got a matrix counterpart (respectively ./ , .* and .^), which apply the operation element by element. For instance, compare: x*y and x.*y.
Type:

```
>> x^2
>> x.^2
>> x.*x
>> x*x
```

if you type ">>whos" now, you will see that you have a lot of variables in the workspace. Type ">>clear", and ">>whos" again….

Matrices operators:

Type

A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]

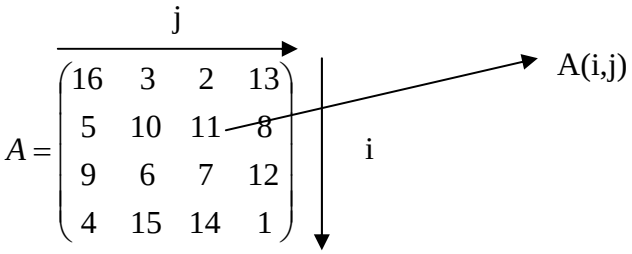
sum(A) % displays the sum of the columns of A.

>>sum(A)
>> 34 34 34 34
sum(sum(A)) % dsiplays the sum of A

A' % Transpose conjugate (if complex number) else transpose.
>> A'

>>sum(A') % Sum of the lines of A (columns of A'). Note that functions can be
% combined very easily.

subscripts:



A(4,2)?

try A(4,5)

ERROR: index exceeds matrix dimension!

The functions available and the toolboxes

Type:
>> help elfun % elementary functions
>> help specfun % special functions
>> help elmat % elementary math functions

to see some of what's there. Try out some of the functions.

For instance, type:
>> t =-pi:0.01:pi;
>> c = cos(t);
>> s = sin(t);

Important toolboxes are available in EPS:

Image processing toolbox.	**>> help images**
Signal processing toolbox.	**>> help signal**
Statistics toolbox	**>> help stats**

Neural networks toolbox >> help nnet

Plotting functions:

Type

```
>> figure
>> plot(s)
>> figure
>> plot(t,s)
>> figure
>> plot(t,s,'g',t,c,'r')
```

Type help plot if you haven't already done it to see more possibilities.
Try and modify the axis, put a title and labels on the x- and y- axes…

Add a title, a label on the x-axis.

Image processing:

Do

```
im1 = imread('peppers.png');      % Reads and image and puts it in im1
im2 = randn(384,512) ;            % Creates a 384x512 normally distributed image
                                  % (gaussian noise) of the size of im1.
whos                              % Note that images are stored in uint8 to save memory
                                  % space
```

Do

```
imshow(im1);                      % Display matrices as images
imshow(im2);
```

Do

```
figure(1)
imshow(im1);                      % Display matrices as images
figure(2)
imshow(im2);                      % Same thing with 2 windows.
```

Play with the figure menus and especially the export button that allows saving images and figures. Type help figure for more information.

Do

```
subplot(2,1,1)
imshow(im1);                      % Display matrices as images
subplot(2,1,2)
imshow(im2);                      % Same thing with 2 images in one window.
```

Do

```
clf;                              % Clear figure
```

```
imshow(im1(:,:,1))          % Displays R component of the image
```

Do

```
clf;
subplot(3,1,1)
imshow(im1(:,:,1))          % Displays Red component of the image
title('Red Component');
subplot(3,1,2)
imshow(im1(:,:,2))          % Displays Green component of the image
title('Green Component');
subplot(3,1,3)
imshow(im1(:,:,3))          % Displays Blue component of the image
title('Blue Component');
```

BE CAREFUL, OPERATORS DO NOT WORK ON UINT8 TYPE

Do

```
im1 = rgb2gray(im1); % Converts image to grey level image an
```

Operators and images:

Addition:

Do

```
im1 = double(im1);  % Converts to double to enable operators use.
ima = im1 + 50*im2;
imshow(ima);    % Does not work all the time
%USE INSTEAD:
imshow(uint8(ima));  % Crops values between [0,255]
%OR
imagesc(ima);axis image; %Recommended solution
```

Substraction:

Do

```
ima = im1 – 50*im2;
imagesc(ima);axis image;
```

Multiplication:

Do

```
ima = im1*im2;
```

This is a matrix multiplication. Cannot be applied to multiply images.
The correct way to do this is:

```
ima = im1 .* im2;
imagesc(ima); axis image;
```

Other operators

/ Matrix division
^ Power operator
‘ Complex transpose

The combination of . and any other operator makes it an element to element operator. For instance .* or ./ multiplies and divides each element of the first image by the element of the second.

Workspace and saving results:

As matlab is interpreted, the data are always in the workspace and can be saved. The workspace can be seen as an area of memory accessible from the command line.

To see it: **whos**

To clear it: **clear**[2]

To save it: **save name** Do help save to learn more about save.

To load it **load name** Do help load to learn more about load.

YOU CANNOT SAVE ANYWHERE. YOU NEED TO HAVE WRITE PERMISSION.

To know where you are, use pwd.
To change location use cd path or cd(‘path’)

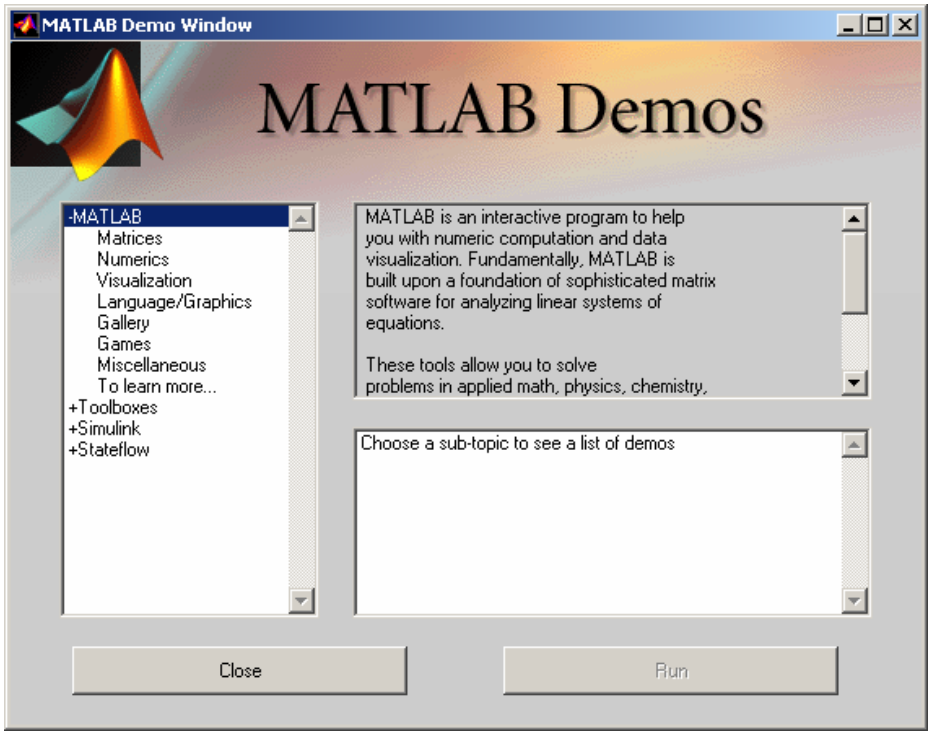
Do

save trial	Comments?
cd H:; pwd;	
save trial;	Comments?
clear all;	
whos;	
load trial	
whos	

How to run the demo

Use the **demo** tool typing demo in the command line. You should get the following window.

[2] clear a clears the variable named a in the workspace. clear all clear everything.



Today or/and during your (precious) spare time, run and try the various demos. I especially recommend to try the first 4 and the image processing toolbox in toolboxes.
This can be used as a self-training tool very effectively.

Conclusion:

Well, that's you started…Now, hopefully, matlab should feel more like an help to investigate, understand and solve mathematical and image processing problems. It is also a great tool to visualize data. You will need to use it to complete your assignments. You will also need to spend a bit of time to fully discover matlab but the reward is huge.
Go to the next page for a memento of where you can find help.
In the next session, we will see how matlab can be used as a programming language.

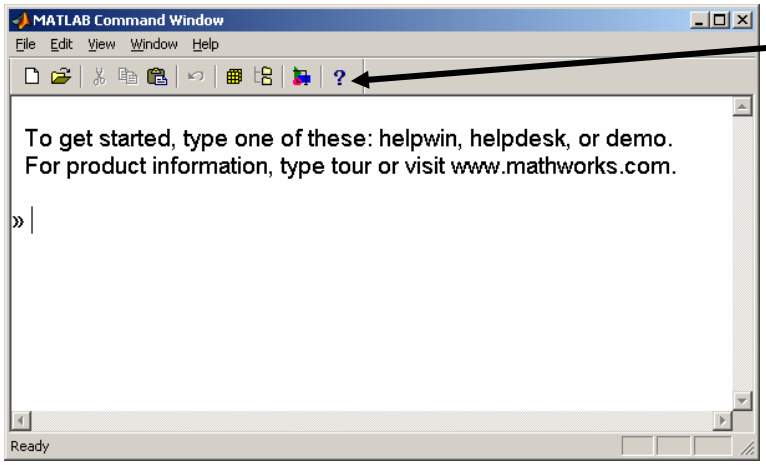
How to start Matlab:

Press the matlab Icon on your desktop.

How to know where you are:

use the pwd command to know where you are
use the dir command to list the files in your current directory
use the cd command to change directory

How to get help:



Press this button

Or type help in the command line

Or type lookfor ‘string’ in the command line

Or type Matlab help desk in the help window

Or go to http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml

How to get an overview of matlab:

Use the **demo** tool typing demo in the command line. This can be used as a self-training tool.

How to see what’s in your workspace:

```
>> who
>> whos                % more detailed
```

How to save and load a workspace

use the save and load commands to save results and data.
see help save for more details.