HERIOT-WATT UNIVERSITY

DEPARTMENT OF COMPUTING AND ELECTRICAL ENGINEERING

B35SE1 – Matlab tutorial 2
MATLAB PROGRAMMING

Objectives:

Remember last week? Well you'd better.
We have introduced you to the basic use of matlab as a program to perform on-line simple calculations, analysis and display. But it is much more than a glorified calculator as we will see today.

During this session, you will learn:

1- Basic flow control and programming language
2- How to write scripts (main functions) with matlab
3- How to write functions with matlab
4- How to use the debugger
5- How to use the graphical interface
6- Examples of useful scripts and functions for image processing

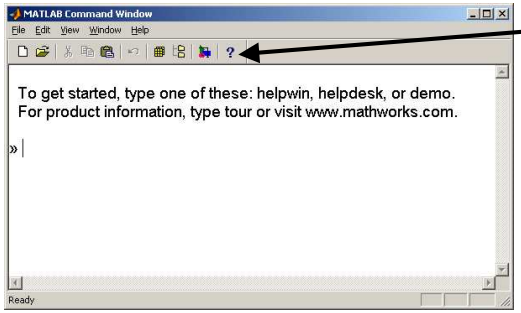
After this hour, you will know most of what you need to know about Matlab and should definitely know how to go on learning about it on your own. So the "programming" aspect won't be an issue any more, and you will be able to use matlab as a tool to help you with you math, electronics, signal & image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

Next page is a reminder of last week main points:

How to start Matlab:

Press the matlab Icon on your desktop.

How to get help:



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.

Matlab ressources:

- Language: High level matrix/vector language with
  - Scripts and main programs
  - Functions
  - Flow statements (for, while)
  - Control statements (if,else)
  - data structures (struct, cells)
  - input/ouputs (read,write,save)
  - object oriented programming.
- Environment
  - Command window. You are now familiar with it
  - Editor
  - Debugger
  - Profiler (evaluate performances)
- Mathematical libraries
  - Vast collection of functions
- API
  - Call C functions from matlab
  - Call Matlab functions form C programs
- **GUI tool**
  - Allows to design easily Graphical User Interfaces to your programs.

Scripts and main programs

In matlab, scripts are the equivalent of main programs. The variables declared in a script are visible in the workspace and they can be saved. Scripts can therefore take a lot of memory if you are not careful, especially when dealing with images. To create a script, you will need to start the editor, write your code and run it.

How to use the editor

Just type edit in the command line and the editor starts.
This should start the editor with a new file. Write the following in the file and save the file as readimage.m in your local directory. To go to your local directory, ude the cd command as seen last week.

```
% This is my first script in matlab
% This script reads in an image, displays it and changes the colormap

% First create a sine wave at a given frequency
Fs = 1000;      % This is the sampling frequency
Ts = 1/Fs;      % sampling interval
t = 0:Ts:1;     %sampling points
w = 10;         % Sine frequency
y = sin(2*pi*w*t);   % Create sine wave

% Now display it
clf;            % Close figure 1
```



```
figure(1);      % First create a figure
plot(t,y,'-m');  % Plot the sine wave in continuous magenta color
```

Scripts and Functions:

You've already written your first elementary script. Standard branching and looping instructions can be used as well: "for" loops, "if" statements etc… To find out about their syntax, type "help" for these different instructions.

<u>if statement:</u>

the if statement is written as follows:

if (*condition1*)
    expression1
elseif (*condition2*)
    expression2
else
    expression3
end

<u>Logical operators:</u>

A==B	**equality for scalars**
isequal(A,B)	**equality for vectors, matrices, structures**
A~=B	**difference**
A>B	**superior. Will return an array of boolean for arrays with 1 for elements where the condition is satisfied and 0 elsewhere.**
A<B	**inferior. Will return an array of boolean for arrays with 1 for elements where the condition is satisfied and 0 elsewhere.**
A>=B	
A<=B	
isempty(A)	**Check if the variable exists and is not empty.**
isinf(A)	**Returns 1 if A is Inf (infinity), 0 otherwise.**
isnan(A)	**Returns 1 if A is not a number (NaN), O otherwise[1].**

<u>Switch and case:</u>

switch (*expression*)
    case *value1*
        action1;
    case *value2*
        action2;
    otherwise
        default action;
    end

[1] Note that NaN and Inf are defined. Example A = Inf

For loops[2]

```
for index = 1 : m
        r(index) = index;
end
```

Example:

```
Do
        sine = zeros(1000,1);
        for i = 1:1000
                sine(i) = sin(2*pi*10/1000*i);
        end
        plot(sine)          % imshow is buggy and sometimes does not work
                            % properly. In that case use imshow then imagesc or
                            % directly imagesc.
```

while loops[3]

```
while (expression)
        Action
end
```

Functions

Functions differ from scripts in that they take explicit input and output arguments. Type "help" function to know more and see examples. As all other matlab commands, a function can be called within a script of from the command line.

The synthax of a function is as follows:

***function* [out1,out2,...,outn] = *function_name*(in1,in2,...,inn)**

Tips:

nargin: **gives the number f input arguments**
allow defaults parameters
allow variable number of arguments

Debugging a program

Debugging is easy in matlab as it is an interpreted language. In order to start the debug mode, you just have to type in the command line:

```
>>dbstop if error          % Will stop for errors
>>dbstop if warning        % Will stop for warnings
```

Alternatively, you can go into the editor and set the options and breakpoints where you want. In any case, the editor will be invoked and show the line where the error has occurred.
In this case the K symbol appears in the command line.
As matlab is interpreted you can change the values of variables and continue the execution using:

[2] They are slow in matlab and can sometime be avoided by vectorizing your programs.
[3] They are slow in matlab and can sometime be avoided by vectorizing your programs.


```
K>>dbstep          %continues by one line
K>>dbcont          % continues until next stop
```

To see all the options of the debugger, use help dbstop.

To quit the debugger, use dbquit
To remove breakpoints and debugger options, use dbclear

Finally you can use the keyboard functions. This does not start the debugger but stops the execution where the keyboard has been typed. You can see the variables and resume execution typing dbcont.

Function/Command Duality

Command line	In function or script
load August17.dat	load('August17.dat')
save January25.dat	save('January25.dat');
cd H:\matlab\	cd('H:\matlab')

General Tips

Vectorization:

```
x = 0;
for k = 1 : 1001
  y(k) = log10(x);
  x = x +0.01;
end
plot(x,y);
```

```
x = 0:0.01:10;
y = log10(x);
```

10 times faster!

Preallocation:

```
r = zeros(32,1);
for n = 1 : 32
  r(n) = n;
 end
```

← **5 times faster!**

Matrices

zeros(m,n) **Creates a mxn matrice filled with zeros**

ones(m,n)	**Creates a mxn matrice filled with ones**
rand(m,n)	**Creates a mxn matrice following a uniform distribution in [0,1]**
randn(m,n)	**Creates a mxn matrice following a normal gaussian distribution (mu = 0, sigma = 1)**