Introduction to MATLAB

Objective: The objective is to get acquainted with MATLAB Material: From CampusNet you can download the following literature and `.m`-files.

- Edward Neuman: *Getting started with MATLAB*, Southern Illinois University at Carbondale.
- David F. Griffiths: *An Introduction to MATLAB*, The University Dundee, 1997.
- Edward Neuman: *Programming in MATLAB*, Southern Illinois University at Carbondale.
- Edward Neuman: *Using MATLAB in Linear Algebra*, Southern Illinois University at Carbondale.

Preparation: None MATLAB Help: MATLAB help is obtained by typing `helpdesk` at the MATLAB command prompt or by typing `help <function name>` in the command prompt. In practice, the fastest and easiest way to get help in Matlab is often to simply Google your problem. For instance: `"How to add legends to a plot in Matlab"` or the content of an error message. In the later case, it is often helpful to find the *simplest* script or input to script which will raise the error.

1.1 Installing the 02450 Toolbox

The course will make use of several specialized scripts and toolboxes not included with Matlab. These are distributed as a toolbox which need to be installed.

1.1.1 Download and unzip the 02450 Toolbox for Matlab, `02450Toolbox_Matlab.zip` (available from Campusnet). It will be assumed the toolbox is unpacked to create the directories:

```
<base-dir>/02450Toolbox_Matlab/Tools/     # Misc. tools and packages
<base-dir>/02450Toolbox_Matlab/Data/      # Datasets directory
<base-dir>/02450Toolbox_Matlab/Scripts/   # Example solutions.
```

We recommend storing your own scripts in a subdirectory of `<base-dir>/02450Toolbox_Matlab/` since this will make it easier to load data and compare your solution to those found in `<base-dir>/02450Toolbox_Matlab/Scripts/`. For instance the solutions to exercise 1 of week 1 could be placed in the file:

```
<base-dir>/02450Toolbox_Matlab/week1/exercise1.m
```

1.1.2 To finalize the installation you need to update your path. To do this run the file `<base-dir>/02450Toolbox_Matlab/setup.m` and ensure you do not get any errors.

1.2 Getting started with MATLAB

1.2.1 Carry out the tutorials by Edward Neuman *Getting started with MATLAB* as well as David F. Griffiths *An Introduction to MATLAB*.
(You can also watch all the Matlab video tutorials (about 50 minutes of video) and carry out the operations performed in the tutorials. You can watch the tutorials by clicking on *Help* in the menu and select *Demos*.)

Make sure you understand the following steps

1.2.2 The *colon* notation is very important in MATLAB as it avoids loops, which run very slowly in MATLAB. Type `help colon` in the MATLAB command prompt for help on the colon notation.

Generating vectors in MATLAB is easy when using the colon notation. Observe the results produced in MATLAB when typing

```
x = 0:6
x = 2:4:17
x = 100:-1:95
x = 1.2:0.1:1.9
x = pi*[0:0.5:2]
```

Extracting the elements from vectors is easy. Consider the following definition of `x` and the echoed results of the last four lines.

```
x = [zeros(1,2),linspace(0,3,6),ones(1,3)]
x(2:5)
size(x)
length(x)
x(2:2:end)
```

Inserting numbers into vectors is also easy. Using the same definition of `x` and observe the results when typing

```
y = x;
y(2:2:end) = pi
y(2:2:end) = 2:2:10
```

Observe the results when indexing the vector `y` with `y(1)` and `y(0)`. Is `y(0)` defined?

1.2.3 Multiplication of vectors in MATLAB may be done element wise or as vector multiplication. Observe the following results when multiplying the vectors `x` and `y` together.

```
x = 1:5
y = 2:2:10
x.*y
x*y'
```

The `'` is the transpose operator. The array multiplier `.*` computes an element wise multiplication of the vectors. In general, the dot `.` indicates array arithmetic operation, see e.g., `help power` and `help rdivide`. The `*` operator is the matrix multiplication.

1.2.4 MATLAB has a built-in editor, which may be used to write *scripts* and *functions*. Start the editor in MATLAB by typing `edit`. Write the following lines in the editor

```
clear
x = 1:9;
whos
```

and save the file as `myscript.m`. Run the script in MATLAB by typing `myscript` at the command prompt and observe the results.

Make a new file `myfunction.m` and write the following function

```
function x = myfunction(n)
%---------------------
% THE MYFUNCTION HELP
%---------------------
x = 1:n;
```

Note that the percentage sign `%` indicates comments and the rest of the line is not evaluated. Run the function in MATLAB with `y = myfunction(9)` and observe the `y` variable. Write at the prompt `help myfunction` and observe the results. Compare the variables `x` and `y`.

1.2.5 MATLAB may be used for plotting results. Write the following lines in the script `myplot.m`.

```
figure(1)
x = 0:0.01:1;
f = exp(x);
plot(x,f);
xlabel('x')
ylabel('f(x)=exp(x)')
title('The exponential function')
```

Run the script and observe the results. The `plot` function creates a simple linear plot. Try plotting other function like `sin` and `cos`. MATLAB supports a number of 2-D and 3-D plots, see `help graph2d` and `help graph3d` for details.

1.3 OPTIONAL

Carry out the remaining MATLAB tutorials, in the order that they are listed on the first page.