EEE115 Systems Engineering
A concise introduction to Matlab

1. Preliminaries

Matlab is an interpreted programming language for technical computing. Like all programming languages it supports a number of data types and functions, all of which have certain rules and syntax. In this document you will be introduced to a some of these concepts and to simplify your understanding the Matlab commands as you enter them on the command-line will be written in the `Courier` font. You should remember that most of Matlab's internal functions are documented and you can read this by typing `help commandname`. You should also remember the percent symbol, %, can be used to comment out lines in your Matlab program. If a command takes too long to execute or you want to halt a program you have written press the key combination Ctrl+C.

2. Data types

The most common data type is the matrix (Matlab stands for Matrix Laboratory). In Matlab a matrix can be the familiar mathematical entity we all know or it can be viewed as a container for data somewhat similar to an array. Matlab supports many other data types including object oriented programming, structures and cells but we will be concentrating on matrices.

2.1 Creating matrices part 1

```
a=2 % creates a 1x1 matrix or a scalar
b=[1 2] % creates a 1x2 row vector
```

Note the square bracket denotes creating a matrix

```
b1=[1,2] % also creates a 1x2 row vector
```

A comma or space can be used to separate columns

```
c=[1;2] % creates a 2x1 column vector
d=[1,2;3,4] % creates a 2x2 matrix
```

2.2 Manipulating matrices

d(r,c) allows us to access the element located on the row r, column c. Note round brackets are now used.

```
d(2,1) % gives the result 3
d(1,2)=1 % changes element located at 1,2 to 1
```

Matlab can be considered to be a calculator providing functions to manipulate matrices

```
a*d % multiplies matrix d by a
b*c % multiples matrix b by c
c*b % what does this do?
```

Matlab obeys the commuting laws

```
b*d % this works
d*b % this does not
d+[b;c']
```

Creating new matrices

```
e=[c,d] % creates a new 2x3 matrix
f=[b;d] % create a new 3x2 matrix
```

Note the use of a comma in the column dimension and semicolon in the row direction

```
g=b' % transpose of b
```

```
h=[d,g]
```

To change an entire column or row we can use the range feature of Matlab denoted by the colon

```
h(:,1)=2 % sets all elements in column 1 equal to 2
h(2,:)=h(1,:) % sets row 2 equal to row 1
```

2. 3 Creating matrices part 2

Matlab has a number of facilities and functions for creating matrices with specific properties. To create a sequence of number from 1 to 10 you could use

```
s1=1:10 % creates a sequence of numbers from 1 to 10
```

This concept can be extended to creating numbers over a range with a step

```
s2=1:2:20 % creates a sequence of numbers from 1 to 20 in steps of 2
s3=1e3:1e2:1e4 % creates a sequence of number from 1000 to 10000 in steps of 100
```

Notice that Matlab accepts engineering notation for numbers. You also cannot have failed to notice a large number of results were generated. To suppress the display we can end a line with a semicolon

```
s4=2*(1e3:1e2:1e4); % creates a sequence of number from 2000 to 20000 in steps of 200
```

In fact the semicolon can be used to separate commands on a single line as follows

```
s5=[1,2];s6=[3;4];s5*s6
```

To create a sequence of 15 numbers equally spaced in the range 20 and 200 we can use

```
s7=linspace(20,200,15)
```

To create a sequence of 25 numbers logarithmically spaced in the range 10 and 10000 we can use

```
s8=logspace(2,4,25) % 10^2=100, 10^4=10000 define the range
```

Special matrices

eye(r,c) generates an rxc identity matrix

```
eye(3) % creates 3x3 identity matrix
```

ones(r,c) generates a matrix with all elements set to 1

```
ones(1,10) % creates a 1x10 row vector with all elements set to 1
```

zeros(r,c) generates a matrix with all elements set to 0

```
zeros(10,1) % creates a 10x1 column vector with all elements set to 0
```

rand(r,c) generates a rxc matrix containing rxc uniformly distributed random numbers in the range 0 to 1

```
r=10*(rand(100,1)-0.5)+5; hist(r) % what does this do?
```

The function randn generates normally distributed random numbers similarly

3. Useful functions and data plotting

Finding the minimum, maximum and mean of a matrix.

```
min(r)
max(r)
mean(r) % are these values what you expected?
```

Use the `whos` command to display all the variables you have created in the workspace. What is the

variable `ans`?
You can clear variables from the workspace using the clear command
```
clear s8 % removes variable s8 from memory
```
Now clear all the workspace using
```
clear all
```

Use the following command sequence to generate a sine wave
```
t=0:0.01:10;y=10*sin(2*pi*t);
```
The sinewave can be plotted using
```
plot(t,y)
```
Axis labels and a title can be added using
```
xlabel('Time (s)');ylabel('Amplitude (V)');title('1Hz sine wave')
```
Now generate a 2 Hz cosine wave with an amplitude of 3 and store it a matrix called z
This can now be plotted on the same figure using
```
hold on;plot(t,z,'r') % the ,'r' changes the line colour to red
```
The figure annotation can now be tidied up by changing the title and adding a legend
```
title('Waveforms');legend('1Hz sin','2Hz cos')
```

Now let's plot the waveforms in the same figure but on difference axes.
```
figure % create new figure
subplot(2,1,1) % create a figure containing two axes one above the
other and make axis 1 active
plot(t,y)
subplot(2,1,2) % make axis 2 active
plot(t,z)
xlabel('Time (s)');ylabel('Amplitude (V)');title('2Hz cos wave')
subplot(2,1,1) % go back to axis 1
xlabel('Time (s)');ylabel('Amplitude (V)');title('1Hz sin wave')
```

The concepts of annotation and subplots can be applied to plotting function such as the hist command we used earlier

4. Writing your own program or script

Programs are useful if have perform a number of operations within a loop (i.e. `for` or `while`), have some decision making to do with your data (i.e. `if else`) or just have too many steps to go through. Matlab contains its own text editor which can be invoked using the edit command. To generate a program to run the subplot example described earlier type
```
edit splot1
```
The editor should have loaded. Now copy and paste the t,y and z vector definitions and then the subplot example from the command history. You program should look exactly like what you type into the command line so you will have to manually delete any added characters. Then click save. You can run your script using
```
splot1
```
A new figure should appear. You can now copy this figure into your favourite word processor. To make your figure look effective you should choose to copy them as an enhanced metafile, a scalable format that should not loose resolution. If you are producing a poster the figure can sometimes look a little 'thin'. The figure can be 'fattened' by selecting PowerPoint from copy options which are buried away in figure menu system. Rather than copying you can also save or export the figure as

some form of picture. Try out these options.