IMAT5119: Matlab Laboratory 2 [1]

Learning Outcomes

You will learn how to access columns and rows of data, how to do multiple plots, and how to use some of the Matlab control structures. You will meet the idea of an m-file and will also be able to draw some membership functions.

Tasks for Lab 2

Use the *load* command to get the data from the file data.txt that you created last time into a matrix called **Data**. Check what is in **Data**.

```
>> Data = load('data.txt');
```

Get the second column of **Data**:

```
>> Data(:,2)
```

Now get the second row:

```
>> Data(2,:)
```

Set the vector **a** to be the first column of **Data** but don't bother to display again:

```
>> a = Data(:,1);
```

Add a column to **Data**.

```
>> Data = [Data [1;2]]
```

(I picked the numbers 1 and 2; you can pick any numbers.)

Matlab has standard programming features like for loops:

```
>> for i = 1:2
       f = 0.05*randn(size(a));
       Data(:,i+1) = Data(:,i+1)+f;
   end
```

f is a column of random numbers; this column is multiplied by 0.05. **f** is the same size as **a**, which is just one of the columns from **Data**. We then add **f** to column $i+1$ for $i = 1$ and then for $i = 2$. Hence this piece of code has altered the 2nd and 3rd columns of Data by adding random elements to them.

There are also if statements etc. which you can use — see the help for syntax.

[1]Sarah Greenfield (acknowledging Bob John, Paco Chiclana and Jenny Carter), 16/10/20

Before you proceed save your new version of Data in data.txt — you will be using this data later on.

```
>> save data.txt Data -ascii
```

We can plot all the columns against the x values (i.e. the values in column 1 of the matrix **Data**) on one graph by typing this,

```
>> plot (Data(:,1),Data)
```

or we can use the figure command to get a new figure and plot on that.

```
>> for i=1:3
      figure
      plot(Data(:,1),Data(:,i))
   end
```

m-files

An m-file allows you to repeat a series of commands that you have created without having to type them all in again. This is particularly useful for conducting experiments where you want to repeat tests with only minor changes. All Matlab commands are m-files.

Create a new m-file by choosing File—New from the menu and put the following commands in:

```
a = 0:0.1:pi;
c = a.^2;
plot(a,c,'+:')
```

Save this as myplot.m. You will now be able to enter myplot at the command line and see the effect. (Note that this file must be in the specified path.)

Membership Functions

You have come across membership functions which fully define fuzzy sets. Now let us draw some membership functions. Matlab has a number of membership functions available:

dsigmf – The difference between 2 sigmoidal membership functions (a 'hat' shape);

gauss2mf – Gaussian combination membership function;

gaussmf – Gaussian curve membership function;

gbellmf – Generalised bell-shaped membership function;

pimf – π-shaped curve membership function;

psigmf – The product of two sigmoid functions;

sigmf – Sigmoid membership function;

smf – S-shaped membership function;

trapmf – Trapezoidal shaped membership function;

trimf – Triangular shaped membership function;

zmf – Z-shaped membership function.

The three most commonly used membership functions are gaussmf, trapmf, and trimf.

All of these membership functions take various parameters. To find out more type help followed by the function name. Type in the following code:

```
>> x = 0:0.1:10;
>> y = trimf(x,[3 6 8]);
>> plot(x,y);
>> xlabel('An example triangular membership function')
```

The three parameters (3,6,8) determine the points that at which the triangle joins the x-axis (3,8) and the peak (6). Experiment with some other membership functions -– especially gaussmf and trapmf.

Another useful Matlab command is evalmf which evaluates any membership function. Use the help system to find out how this works.