SCHOOL OF PHYSICS, ASTRONOMY & MATHEMATICS

4PAM1008 MATLAB

4 – Visualising Data

Dr Richard Greenaway

4 Visualising Data

4.1 Simple Data Plotting

You should now be familiar with the plot function which is one of many powerful functions for plotting data only some of which will be covered here. See MATLAB help for a complete guide to all the plotting functions. You can control every aspect of the graph through a system called **handle graphics** but we will not be covering that topic in this workshop. Here we will confine ourselves to high-level plotting and formatting functions

We will now explore the plot command a bit further to introduce some general concepts before introducing some other plotting functions. Below is an example plot. The data is plotted in a window called the figure window. Inside the figure window you can have one or more sets of axes on which you will plot your data.

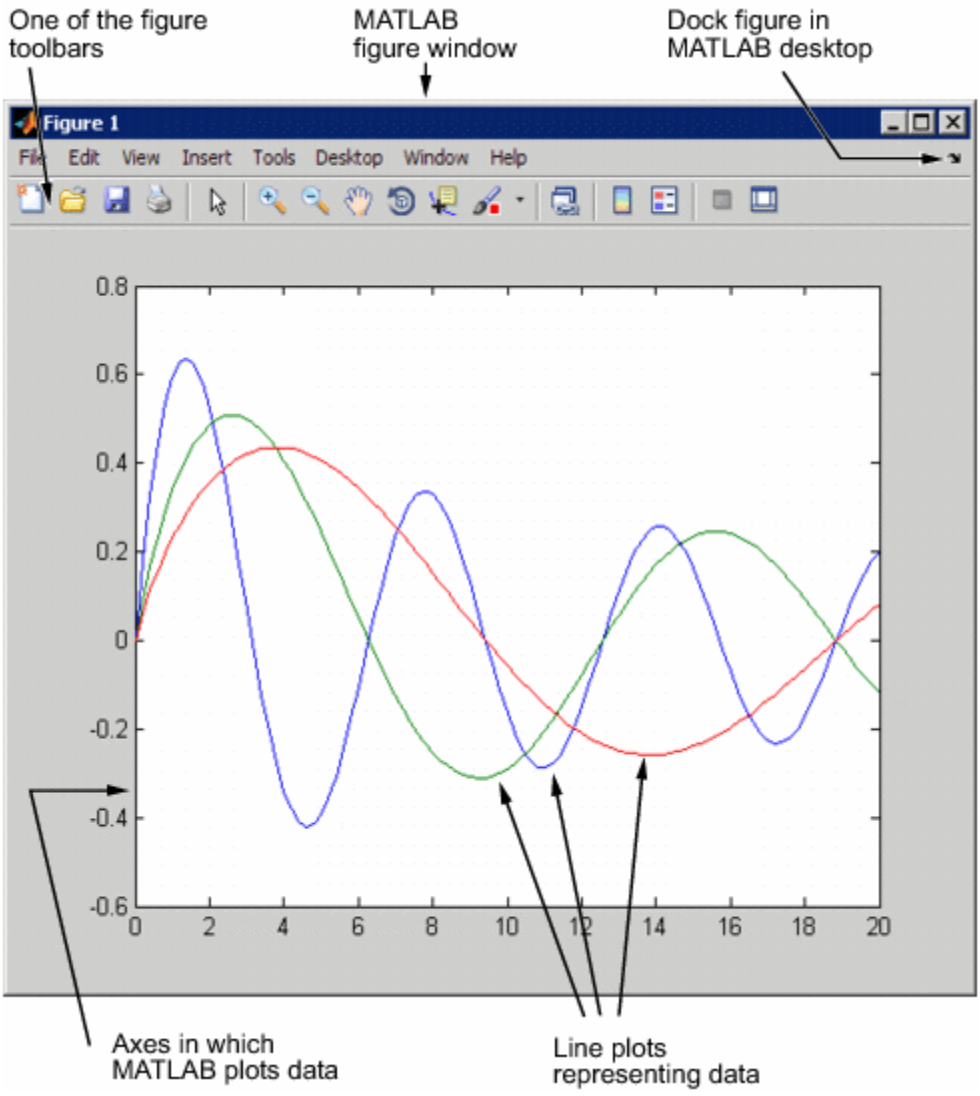


Figure 1 (*reprinted from MATLAB Help*)

The example in Figure 1 contains three different plots on one axis which you have already done in an earlier workshop by plotting 3 data sets from an n x 3 matrix. Let's re-create the figure above, but this time we will plot each data set separately.

First, create the data.

```
>> x = [0:.2:20];
>> y = sin(x)./sqrt(x+1);
>> y2 = sin(x/2)./sqrt(x+1);
>> y3 = sin(x/3)./sqrt(x+1);
```

[data_4_1.mat]

Now plot the first data set, `y` against `x`.

```
>> plot(x,y)
```

Now, if we simply ran `plot` again specifying the second data set, the first plot would be overwritten. Try it!

```
>> plot(x,y2)
```

In order to plot data `y2` while maintaining the `y` data plot we need to use the command `hold`. So, re-generate the first plot, and then add the second as follows.

```
>> plot(x,y)
>> hold on
>> plot(x,y2, 'g')
```

Notice the extra parameter when `plot` is called for the second time. `plot` can take any number of extra parameters which alter the various characteristics of the plot. These optional parameters usually take the form of parameter-pairs. The first identifies the property to set and the second the new 'value' that the property should have. The above example is actually short for.

```
>> plot(x,y2, 'color','g')
```

The property `'color'` refers to the line colour (yes, MATLAB uses American spelling, curse them) and `'g'` stands for green. However, plot allows you to use a short hand for some of the basic properties.

Try this.

```
>> close
>> plot(x,y2, 'ko:')
```

This is actually short for...

```
plot(x,y2, 'color','k','LineStyle',':','Marker','o');
```

Table 4-1 Lists the various line style symbols available.

Colours		Line styles	
y	yellow	.	point
m	magenta	o	cirlce
c	cyan	x	x-mark
r	red	+	plus
g	green	-	solid
b	blue	*	star
w	white	:	dotted
k	black	-.	dashdot
		--	dashed

Table 4-1 Symbols for Defining Line Styles and Markers

Getting back to the original task, to generate our own version of Figure 1, do the following.

```
>> plot(x,y)
>> hold on
>> plot(x,y2, 'g')
>> plot(x,y3,'r');
```

Notice that we only had to run the hold command once. If you want to stop adding data to the plot then type

```
hold off
```

What do you do if you want to plot the lines on separate graphs. We have already seen that simply calling `plot` again overwrites existing data. Well, we create a new figure window before calling `plot`.

```
>> plot(x,y)
>> figure;
>> plot(x,y2, 'g')
```

The `figure` command opens a new empty figure and when you call `plot` the data will be plotted in that window.

When you have more than one figure window open at once you select a window with the mouse to make it *active* and any commands executed subsequently will 'target' that figure.

4.2 Formatting and Annotating Plots

In this section you will be introduced to a few functions which will enable you to carry out basic formatting of plots - setting axis range, adding titles an such like.

The `axis` command provides a high-level means to control the scaling and appearance of axes. Perhaps the most important, is to set the range of the axes.

Recreate Figure 1 if it no longer exists. The m-file **figure1.m** will do it for you to save time.

```
>> figure1
```

Now we will change the scale of the x-axis to show the plots between 0 and 10 and keeping the y-axis the same.

```
>> axis([1 10 -0.6 0.8]);
```

The format is `axis([ xmin xmax ymin ymax])`. You can also use `axis` to format the axes in various ways. Try the following and observe what happens to the figure.

```
>> axis equal
>> axis normal
>> axis off
>> axis on
```

Having plotted data, we can add titles and labels. Try the following and see how the figure looks afterwards.

```
>> title('An Example Data Plot');
>> xlabel('X-values')
>> ylabel('Y-values')
>> legend('First','Second','Third');
```

4.3 Using `fplot` to Quickly Viualise a Function

So far, in order to visualise a function we have had to generate example data explicitly. However, the function `fplot` provides a simple way to plot a function between specified limits without having to generate values first.

`fplot` can be used to plot any function of the form $y = f(x)$ between specified limits.

```
fplot(function, limits)
```

The first parameter is the function to plot which you can specify in various ways, the simplest of which is as a string. You can specify x-limits only as a 1 x 2 vector (`[xmin xmax]`) or x and y-limits as a 1 x 4 vector (`[xmin xmax ymin ymax]`). So for example:

```
fplot('sin(x)',[-2*pi 2*pi]);
```

Exercise 4-1

1. Generate a vector *x* values between 0.01 and 0.1
2. Now calculate the function $y = \sin\left(\frac{1}{x}\right)$
3. Now use `plot` to plot the results.
4. Now generate the same plot using `fplot` **in a separate figure window.**

Compare the two graphs. What differences are there? You will see that `fplot` *does a better job than* `plot`*. This is because* `fplot` *generates x-values at varying intervals depending on how rapidly the function is changing to ensure that there is a smooth plot of fast-changing functions.*

4.4 More Advanced Plots

In this section you will be introduced to a few of the more useful plotting functions, but there are many more which you can find out about in the MATLAB documentation.

4.4.1 Log Plots

If you need to plot data logarithmically then you can use the `semilogx`, `semilogy` and `loglog` functions. These are special versions of plot which display one or both axes as $\log_{10}$ scale.

Exercise 4-2

1. Generate data for the function $y = e^x$ for *x* between 0 and 4 in steps of 0.01.
2. Plot the function using `plot`.
3. In a separate figure plot the same function using `semilogy`.
4. Add grid lines by typing: `grid on`

4.4.2 Error Bar Plots

The `errorbar` function allows you to plot x-y data and associated errorbars. The function behaves like `plot` but takes an extra parameter.

```
errorbar(x,y,e)
```

where `e` is a vector of the same length as `x` and `y` specifying the size of the error bar associated with each x,y data point.

Exercise 4-3

1. Create a data set $y = e^x$ for x between **0** and **1** in steps of **0.1**

2. Now we will create some 'errors' for the associated data using the random number generator `rand`. Type the following.

```
e = 0.5*rand(1,length(x));
```

   where x is your x-vector data created in step 1.

3. Now plot the data as follows

```
errorbar(x,y,e,'o');
```

4.4.3 Bar Plots

Bar plots can be used to represent x-y data and other types such as histograms showing the frequency distribution of values.

Exercise 4-4

1. Generate a set of random numbers, normally distributed about zero using the `randn` function as follows:

```
r = randn(10000,1);
```

We are now going to use the `hist` function to generate a histogram of the data showing the distribution of values in the set.

2. Define the 'bins' for the distribution.

```
x = -4:0.2:4;
```

   That is, we will bin the values from -4 to +4 in steps of 0.2.

3. Now we use `hist` to bin the values.

```
n = hist(r,x);
```

   where `n` is a vector containing the frequency of values centred on each bin value in `x`.

4. Finally plot the histogram.

```
bar(x,n)
```

The function `hist` *will actually plot the data for you if you do not include the return variable in step 3.*

4.4.4 Polar Plots

Of course, functions are not always of the form $y = f(x)$. Some functions are represented in terms of polar coordinates r, θ.

Consider the following function.

$$r = \sin 2\theta \cos 2\theta$$

We could convert this function to Cartesian coordinates.

By simple trigonometry,

$$x = r\cos\theta = (\sin 2\theta \cos 2\theta)\cos\theta$$

$$y = r\sin\theta = (\sin 2\theta \cos 2\theta)\sin\theta$$

Now we could define a vector of θ values and calculate the respective values of *x* and *y* and plot them using `plot`.

However there is a much easier way, we can use the `polar` function which can be used to plot any function expressed in polar coordinates,

```
polar(theta,r)
```

Exercise 4-5

1. Create a vector of θ values from **0** to **2π** in steps of **0.01**.
2. Calculate values for ***r*** then ***x*** and ***y*** using the equations above.
3. Plot the curve using plot. Add a grid by typing `grid on`.
4. Plot the data in *r*,*θ* form using `polar`.

4.5 Creating Multiple Graphs in the Same Figure

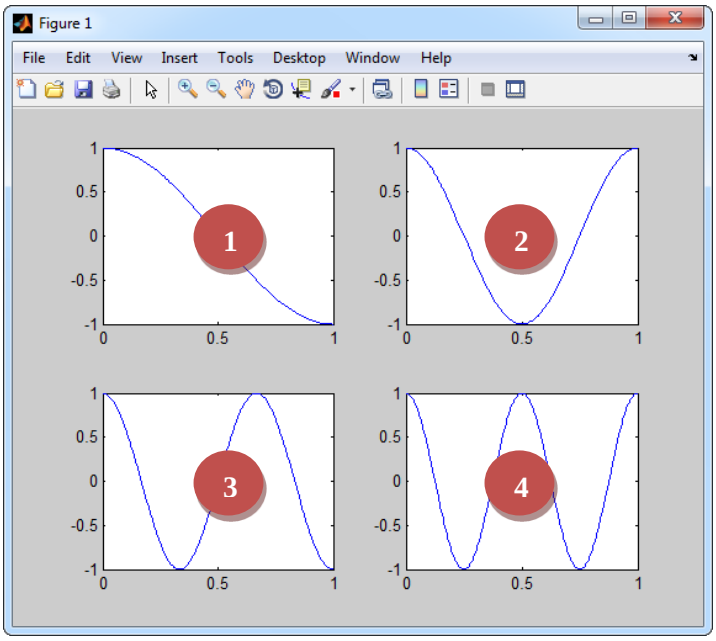
The `subplot` command provides an easy way to have multiple axes in the same figure window. Try the following exercise and watch what happens as each command is executed.

```
>> figure;
>> subplot(2,2,1);
>> fplot('cos(pi*x)',[0 1]);
>> subplot(2,2,2);
>> fplot('cos(2*pi*x)',[0 1]);
>> subplot(2,2,3);
>> fplot('cos(3*pi*x)',[0 1]);
>> subplot(2,2,4);
>> fplot('cos(4*pi*x)',[0 1]);
```

The first line creates an empty figure window. When you call subplot for the first time, a set of axes appear in the top-left quadrant. `subplot` has the following syntax.

```
subplot(m,n,p)
```

`subplot` breaks the figure window into a `m`-by-`n` matrix of axes and makes the p^{th} axis as the current plot. The axes are numbered *rowwise* as shown



Thus, `subplot(2,2,3)` makes the axes in the lower-left quadrant current, and the following call to `fplot` will place the data in those axes. So `subplot` has two functions, if no axes exist it will create them and it will make already existing axes 'active' .

4.6 Saving and Exporting Figures

Any figure can be saved to a file of type **.fig**, which can be reloaded into MATLAB at a later time, simply select *save* from the *File* menu. Additionally you can copy a plot for pasting into documents using the *Edit→Copy Figure* menu item.

4.7 Further Exercises

Exercise 4-6

1) Use `fplot` to visualise the following functions

a) $y = x^2 - 2x - 1, \quad -5 \le x \le 5$

b) $s = \frac{1}{1-t} \quad -6 \le t \le 6, -2 \le s \le 2$

There is a discontinuity at t=1so the default plot (which joins all points by a line) does not work well. Re-do the plot but display the data as a scatter graph (data point markers but no line).

c) $y = x^2 \sin 3x \quad 0 \le x \le 10\pi$