Octave Tutorial 5: How to plot data in Octave

with extracts from *Introduction to Octave*, by P.J.G. Long

In this tutorial you will learn how to

- plot data in Octave.

Octave has powerful facilities for plotting graphs via a second open-source program GNU-PLOT. The basic command is `plot(x,y)`, where `x` and `y` are the co-ordinate. If given just one pair of numbers it plots a point, but usually you pass *vectors*, and it plots all the points given by the two vectors joining them with straight lines.

Let us start with a few examples. First, we will plot the function $f(x) = x^2$ over the interval $[0, 1]$. Here are some instructions on how to do that:

- Create a vector `x` of length 11 containing values between 0 and 1 using the `linspace` command as follows:

  ```
  octave#:#> x = linspace(0,1,11)
  ```

  The vector `x` corresponds to the end points of 10 equally spaced subintervals of $[0, 1]$.

- Now we will map these points to the function $f(x) = x^2$ and store the values in the vector `y`. Enter the following command:

  ```
  octave#:#> y = x.^2.
  ```

  Don't forget the "dot" in the command above, which tells Octave to take the square of each entry of `x`. The command `y= x.*x` also works, but `x*x` will give an error message because the dimensions of the vector `x` do not allow matrix multiplication.

- We now have defined our function on the interval. It's time to plot it! Type in the following command:

  ```
  octave#:#> plot(x,y)
  ```

  You should now have a new window on your screen that contains a plot of $y = x^2$ from $x = 0$ to $x = 1$ using a thin blue line to connect points. This is the default style of plotting in Octave. To change the appearance of the plot, you need to add a third argument to the `plot` command. For example, you can change the plot to appear as a thin red line, with our data points indicated by x's by typing

  ```
  octave#:#> plot(x,y,'r-x')
  ```

  Detailed information on how to change the colour and style for symbols and lines is given below.

In general, the syntax of the `plot` command is `plot(x,y,[options])`. ping in `plot(x,y)` alone, without any `[options]` creates a plot of points connected by a thin blue line, as you saw above. This is the default style of plotting. To change the appearance of plots, there are several options available in Octave. You can change colour, data point markers, line style, etc. The basic options can be implemented as follows:

```
   octave#:#> plot(x,y,'[colour][linestyle][marker]', 'linewidth', [n])
```

where

`colour` : Specifies the colour of the line. Some options are `b,r,k,` or `g`, corresponding to blue, red, black, or green respectively.

`linestyle` : Specifies the style of the line you wish to plot. `-,--` or `(blank space)` are common examples corresponding to solid, dashed, or no line respectively. (Note: the no line option will only work if you specify a marker)

`marker` : Specifies the data marker at each point in your figure. `(blank space), *` or `o` are some examples that correspond to no markers, asterisks, and circles.

`n` : Specifies the thickness of the line being plotted (0.5 is the default).

Note that you can choose to specify values for only some of the options. If any options have unspecified values, they turn to default values. For example,

```
octave#:#> plot(x,y,'ro')
```

will plot `x` and `y` as a series of red circles, not connected by a line.

```
octave#:#> plot(x,y,'b--','linewidth',3)
```

will plot `x` and `y` as a thick, blue, dashed line.

The next example consists in plotting two different functions on the same axis. Specifically, we will plot the functions $f_1(x) = \sin(x)$ and $f_2(x) = \sin(x^2)$ from $x = 0$ to $x = 2\pi$, using equally spaced points on the x-axis. The distance between successive points on the x-axis is set to 0.01 units. Here are some instructions on how to create these plots:

- First of all, we clear the workspace and the figure window. To do that, enter the following commands:

  ```
  octave#:#> clear all; clf;
  ```

  The `clear all` command clears all existing variables and other items from the workspace; it also frees up system memory. The `clf` command clears the current figure window. This practice of clearing the workspace is highly recommended whenever you start a fresh new set of operations in Octave.

- Now create a new vector `x` containing points between 0 and 2π that increase constantly by 0.01. Since you want to control the step size between vector components, the colon operator `:` is a better choice to generate the vector `x` than the `linspace` command, so

  ```
  octave#:#> x = 0:0.01:2*pi;
  ```

- Now define the functions by creating two vectors `y1` and `y2`:

  ```
  octave#:#> y1 = sin(x);
  octave#:#> y2 = sin(x.^2);
  ```

- You can now create the plots by entering the following:

  ```
  octave#:#> plot(x,y1,'k-','linewidth',2);
  ```

  You can add a grid on your plot by typing

  ```
  octave#:#> grid on;
  ```

Now, if you type another `plot` command to plot `y2`, Octave will replace the existing plot with the new, which is not what we want. To overlap the two plot, type

```
octave#:#> hold on;
```

This will allow you to plot multiple graphs within the same figure. From now on, any new graph will be plotted on the same figure (to return to the default mode of one plot per figure, type `hold off` at the end of this example). Now you are ready to type `y2`. Type

```
octave#:#> plot(x,y2,'m--','linewidth',2);
```

You can add a legend by typing

```
octave#:#> legend('f1','f2');
```

Take note of what you see in the plot window. Consider what would have happened if we had not included the `hold on` command.

Here is a recap of the plotting commands used in this tutorial.

`plot(x,y,[options])`: This command will plot corresponding values from vectors `x` and `y` in a figure window. `x`, `y` must be the same length or an error will occur. Typing in `plot(x,y)` alone, without any `[options]` creates a plot of points connected by a thin blue line. Various possibilities exist for the `[options]` This is the default style of plotting. To change the appearance of plots,

`hold on`: This command will allow you to plot multiple sets of data within the same figure, rather than plotting only the last data-set requested. The command `hold off` will turn this feature off.

`grid on`: This command will make a grid appear in the current figure window. Typing `grid off` will remove the grid.

`xlabel('string')`: This command will display the word `string` along the x-axis in the current figure.

`ylabel('string')`: This command will display the word `string` along the y-axis in the current figure.

`title('string')`: This command will display the word `string` as a title to the current figure.

`legend`: Assuming that you have plotted one or more curves in same figure, typing in `legend('string1','string2',...)` will make the legend appear with `string1` corresponding to the first curve that was plotted, `string2` corresponding to the second curve plotted, and so on. Typing `legend off` will remove the legend.

Exercise 1: Plot the following function on the given interval.

$$f(x) = e^{-x^2}, \quad 0 \leq x \leq 1$$

Use 21 points (i.e. 20 subintervals) on the x-axis. Plot it as a solid red line of greater than default thickness. Add a grid and give the plot the title "Gaussian". Recall that you can compute the exponentional function using the pre-defined Octave function `exp(x)` (check out the Octave help for more information on the function `exp(x)`).

Exercise 2 Plot on the same axis as in Exercise 1 the following functions on the interval $0 \leq x \leq 1$,

$$f_1(x) = \sin(2\pi x), \quad f_2(x) = \cos(2\pi x)$$

Let the points on the x-axis be spaced 0.025 units apart. Plot $f_1(x)$ using black asterisks with no connecting line, and $f_2(x)$ using black circles with no connecting line. Label your x- and y-axis, "x" and "y" respectively. Include an appropriate legend.