SCILAB – PLOTS AND GRAPHS

Simple plots

Creating a simple plot in Scilab requires calling a function with 2 vector arguments (either static or resulting from another function), standing for x- and y-values. For example, having some measurement results given in the table below:

x	0,5	0,7	0,9	1,3	1,7	1,8
y	0,1	0,2	0,75	1,5	2,1	2,4

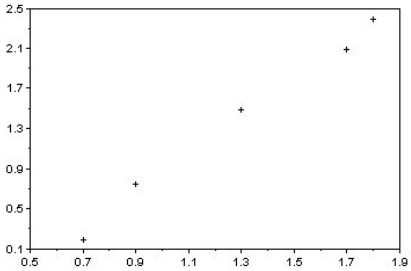
one should define 2 vectors:

```
x = [.5 .7 .9 1.3 1.7 1.8 ];
y = [.1 .2 .75 1.5 2.1 2.4 ];
```

and call the function:

```
plot2d(x,y, style=-1)
```

The graph should now appear (if not, it may be hidden behind other windows). This graph marks the points with an '+':



Other types of points can be used by changing the `style=-1` in the `plot2d` command – use help to find out the details.

Exercise 1

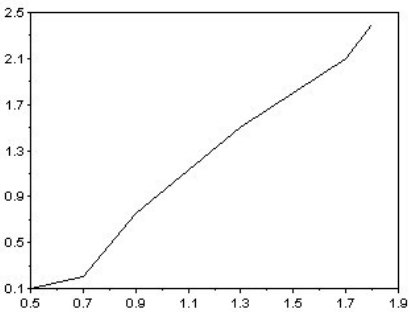
Experiment with different values for the style. Negative values give markers, positive values coloured lines. As you experiment you should erase the previous plot. Use the command:

```
xbasc()
```

Otherwise the plots will overlap (which feature may be desired in other cases). Commands starting with `x` generally are associated with graphics commands (This comes from the X window system used on Unix machines). So `xbasc()`, is a basic graphix command which clears the graphics window.

Very often people use the very simplest command to plot data and automatically type just

```
plot2d(x, y) // The simplest plot command
```



This simple command does not present the data here very well. It is hard to see how many points were in the original data. It is really better to plot just the points, as the lines between points have no

significance; they just help us follow the set of measurements if there are several data sets on the one graph. If we insist on joining the points it is important to mark the individual points as well.

Exercise 2

1. Plot the above data with the points shown as circles.
2. Plot the above data with the points joined by lines (use `help plot2d`).
3. Find out how to add a title to the plot. (use the command `apropos title`)

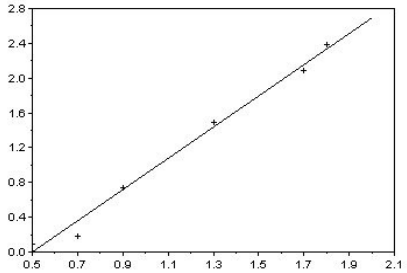
Exercise 3

Change the above plot commands to show the data more clearly by plotting the data points as small circles joined by dotted lines. Use `help plot2d` to see the possible arguments.

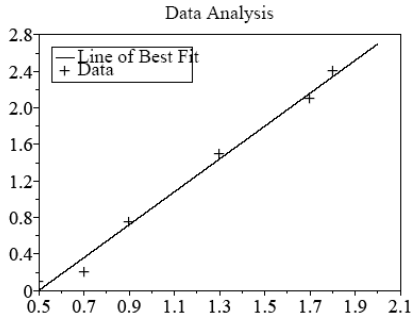
From the graph it is clear that the points almost lie on a straight line. Perhaps the points are off the line because of experimental errors. A course in statistics will show how to calculate a 'line of best fit' for the data. But even without statistics, the line between the points (.5, 0) and (2, 3) is a good candidate for 'a line fitting the data'. Lets add this line to the plot and see how well it approximates the data. We do this by asking Scilab to plot the points (.5, 0) and (2, 2.7) joined by a line.

```
x = [.5 .7 .9 1.3 1.7 1.8 ];
y = [.1 .2 .75 1.5 2.1 2.4 ];
plot2d(x,y,style=-1)
x_vals = [.5 2]; // The X-coords of the endpoints.
y_vals = [0 2.7]; // The Y-coordinates
plot2d(x_vals,y_vals)
legends(['Line of Best Fit'; 'Data'], [1, -1], 5)
xtitle('Data Analysis')
```

Note that `x_vals` contains the x–coordinates, `y_vals` contains the y–coordinates, and the two points are joined by a line because we don't specify a style (default style is a line):



The line of best fit can be found using the `datafit` function, which will not be described in the instruction:



There are many opinions on what makes a good graph. From the scientific viewpoint a simple test is to see whether we can recover the original data easily from the graph. A graph is supposed to add something, not remove information. For example if our data runs from 1 to 1.5 it is a bad idea to use

an axis running from 0 to 10, as we will not be able to see the differences between the values. We give two more subtle recommendations.

In the example in this section, it was easy to see the relationship between x and y from the simple plot of x against y. In more complicated situations, it may be necessary to use different scales to show the data more clearly. Consider the following model results:

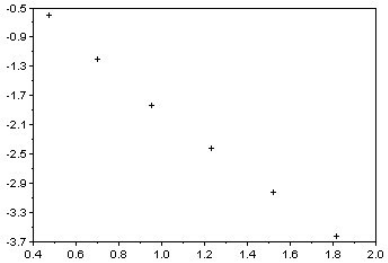
n	3	5	9	17	33	65
s_n	0,257	0,0646	0,0151	$3{,}96\times10^{-3}$	$9{,}78\times10^{-4}$	$2{,}45\times10^{-4}$

A plot of n against s_n directly shows no obvious pattern. (Note the dots, '. . .', in the next example mean the current line continues on the next line. Do not type these dots if you want to put the whole command on the one line.

```
n = [ 3 5 9 17 33 65 ];
s = [ 2.57e-1 6.46e-2 1.51e-2 ...
3.96e-3 9.78e-4 2.45e-4 ];
xbasc() // clear the previous plot
plot2d( n, s, style=-1 ) // This is a poor plot!!
```

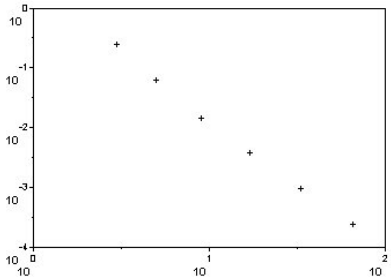
In fact it is hard to read the values of s_n back from the graph. However a plot of $log(n)$ against $log(s_n)$ is much clearer. To produce a plot on a log scale we can use either of the commands:

```
xbasc();
plot2d(log10(n), log10(s), style=-1)
```



or

```
xbasc();
plot2d(n, s, style=-1, logflag = 'll') //see help for 'logflag' option
```



Exercise 4 (10 points)

1. Graph $1/(1+e^{\alpha x})$ for $x\in[-4,4]$ and $\alpha=0.5, 1.0, 2.0$ on the one plot.
2. Use `plot2d` to draw a graph that shows that:

$$\lim_{x\to+\infty}\frac{\sin(x)}{x}=0$$

Select a suitable range and a set of x values.

Printing graphs

When the graph has been correctly drawn we will usually want to print it. As many graphs come out

on the printer, it is a good idea to label the graph with your name. To do this use the `xtitle` command:

```
xtitle('This is my plot') \\ Be sure to put quotes around the title.
```

Bring the plot window to the front. The title should appear on the graph. If everything is fine, click on the 'file' menu → 'print scilab' item in the graph window. If everything is set up correctly, the graph will appear on the printer attached to your computer.

After printing a graph, it is useful to take a pen and label the axes (if that was not done in Scilab), and perhaps explain the various points or curves in the space underneath. Alternatively before printing the graph, use the Scilab command `xtitle` with two optional arguments and the legend argument of the `plot2d` command for a more professional appearance. For instance

```
xbasc()
plot2d(x,[y y2 y3],leg='function y@function y2@function y3')
xtitle('Plot of three functions', 'x label', 'y label')
```

Another option is to use the graphics window file menu item, export, or the `xbasimp` command to print the current graph to a file so that it can be printed later. For example to produce a postscript file use the command:

```
xbasimp(0,'mygraph.eps')
```

This will write the current graph (associated with graphics window 0) to the file *mygraph.eps*. This file can be stored and printed out later.