In this practical we will look at graphics in Scilab. Scilab has a large number of commands for graphics and we won’t be able able to cover them all. Rather we will stick to those that are most commonly used.

## Contents

1. **2D Plotting** 2
   1.1 Simple Plots ........................................ 2
   1.2 Multiple Curves ...................................... 4
   1.3 Multiple Plots ........................................ 5
   1.4 Titles and Captions ................................... 7
   1.5 Other Features ........................................ 8

2. **3D Plotting** 9
   2.1 3D Curves ........................................... 9
   2.2 3D Surfaces .......................................... 10
   2.3 Contour Plots ......................................... 11

3. **Histograms** 12
1 2D Plotting

1.1 Simple Plots

As we have already seen, the simplest type of graph takes two vectors of equal size, and plots one against the other. The vectors may be either row or column vectors, although column vectors are probably preferable as they are used when plotting multiple curves.

```matlab
--> x = (-10:.1:10)';

--> y = sin(x);

--> plot2d(x, y)
```

![Graph example](image)
We can also plot a single vector, whose components are plotted against $1, 2, \ldots, n$ where $n$ is the length of the vector:

```latex
->\text{plot2d}(y)
```

![Graph showing a series of sinusoidal waves with different scaling on the x-axis.](image)

Notice the different scaling on the $x$-axis.

The graphs we have looked at so far were of continuous curves. Data can also be plotted as points by using the `style` option to the `plot2d` command. Negative values for `style` correspond to different types of points, positive values for `style` correspond to different colours. You can use the `xset()` command to bring up a menu with different styles and colours, as well as thing like line thickness.

```latex
->\text{plot2d}(x, y, style = -1)
```
1.2 Multiple Curves

The few times we have plotted multiple curves so far we have just plotted the different graphs without clearing the screen. If in \texttt{plot2d(x, y)} \(y\) is a matrix, then each of the \textit{columns} of the matrix is plotted as a separate curve. In this case \(x\) has to be a column vector. We can also plot the curves in different styles, by setting \texttt{style} to a vector of style numbers.

\begin{verbatim}
plot2d(x, [sin(x) cos(x)], style = [1 -1])
\end{verbatim}
Note in this example that \([\sin(x) \cos(x)]\) is a two column matrix.

### 1.3 Multiple Plots
Multiple graphs can included in one figure using the `subplot` command.

-->`subplot(2, 2, 1)`

-->`plot2d(x, sin(x))`

-->`subplot(2, 2, 1)`

-->`plot2d(x, sin(x))`

-->`subplot(2, 2, 2)`
-->plot2d(x, sin(2*x))
-->subplot(2, 2, 3)
-->plot2d(x, sin(4*x))
-->subplot(2, 2, 4)
-->plot2d(x, sin(8*x))
1.4 Titles and Captions

Titles and captions can be added, after a graph has been drawn, using the \texttt{xtitle} command. It takes three arguments – the title for graph, the caption for the $x$-axis and the caption for the $y$-axis. If you don’t need a title just use a blank string, i.e. ‘’.

\begin{verbatim}
-->x = (-20:0.1:20)’;

-->plot2d(x, sin(x)/x)

-->xtitle('A TITLE', 'x', 'sin(x)/x')
\end{verbatim}

Note the weird placement of the axis labels. There is nothing, at least that I am aware of, that you can do about this.
1.5 Other Features

There are a lot of things Scilab can do with 2D graphs. Here is an example. See the help pages for more information.

--> plot2d(x, sin(x)/x, rect = [-20 -0.5 20 1.5], axesflag = 5, ...
-->   max = [10 4 5 4])

The dots, ..., at the end of the line indicate that the command is going to be continued on the next line.
2 3D Plotting

2.1 3D Curves

Curves in 3 dimensional space can be plotted using `param3d`. It takes three vectors containing the values the $x$, $y$ and $z$ coordinates of the points on the curve. By clicking on the 3D Rot button on the graphics window and playing around with the mouse you can alter the orientation of the graph.

```matlab
-->z = (0:.01:10)';

-->param3d(z.*sin(5*z), z.*cos(5*z), z)
```
2.2 3D Surfaces

The basic command for plotting surfaces is \texttt{plot3d}(x, y, z) Here \(x\) and \(y\) are vectors of \(x\) and \(y\) coordinates of lengths \(n_1\) and \(n_2\) and \(z\) is a \(n_1 \times n_2\) matrix of \(z\) values. We will plot the graph of

\[
z = e^{-(x^2+y^2)}
\]

\[
\rightarrow x = (-3:.1:3)';
\]

\[
\rightarrow y = x;
\]

\[
\rightarrow z = zeros(61, 61);
\]

\[
\rightarrow for \ i = 1:61 \\
\rightarrow \quad for \ j = 1:61 \\
\rightarrow \quad \quad z(i,j) = \exp(- \ x(i)^2 - y(j)^2); \\
\rightarrow \quad end \\
\rightarrow end \\
\rightarrow plot3d(x, y, z)
\]

The \texttt{for} loop in the above example can be a bit slow and is also a bit awkward. An alternative is to use \texttt{eval3d}.

\[
\rightarrow function \ y = func(x, y) \\
\rightarrow \quad y = \exp(- x.*x - y.*y) \\
\rightarrow \quad endfunction \\
\rightarrow z1 = eval3d(func, x, y); \\
\rightarrow max(abs(z1 - z))
\]

\[
\text{ans} = 0.
\]

Note that the function called by \texttt{eval3d} must be able to take vectors as arguments. If we had used \(z = \exp(- x\times x - y\times y)\) we could not use \texttt{eval3d}.  

10
2.3 Contour Plots

If we have a function $z = f(x, y)$ then the level sets of $f$ are the sets $f(x, y) =$ constant. These will, in general, form a family of curves in the $x$-$y$ plane. We can plot these using the command `contour (x, y, z, n)` where $n$ is the number of level sets to be plotted.

```plaintext
-->x = (-10:0.1:10)';

-->y = x;

-->function z = func(x, y)
-->  z = sin(x).*cos(y);
-->endfunction
```
3 Histograms

Histograms can be plotted with the `histplot(n, data)` command. Here `n` is the number of bins in the histogram and `data` is the vector of data for which we want to draw the histogram. The following example draws a histogram of a vector of normally distributed random numbers.

```matlab
--> rr = rand(1, 10000, 'normal');
```
---> histplot(40, rr)