Chapter 6. Graphics in Scilab

Plotting and graphics are among the most effective ways to analyze and present data. Scilab offers powerful built-in tools for creating and customizing different types of plots such as 2D plots, contour plots, and 3D surface plots. In this chapter, we will learn how to generate these basic graphics and how to enhance them with titles, axis labels, and legends for clearer presentation.

1. Displaying 2D and 3D plots:

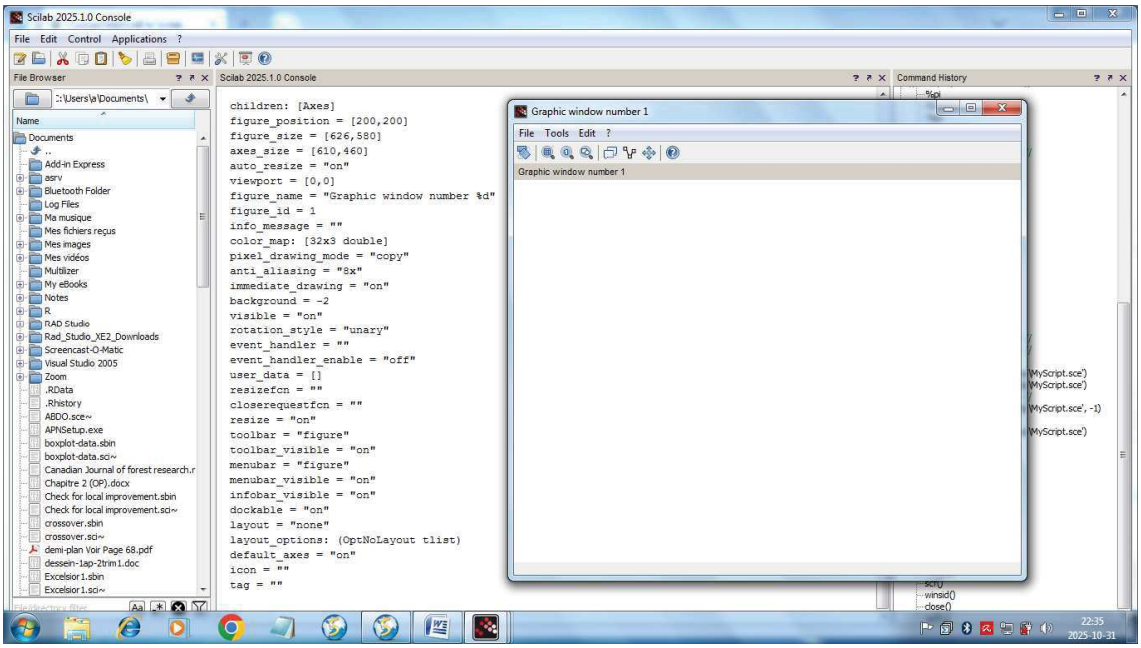
The graphics window opens automatically with plotting commands. To open one manually, use **`scf(number)`**:

```
-->scf(1)
```

or

```
--> figure(1).
```

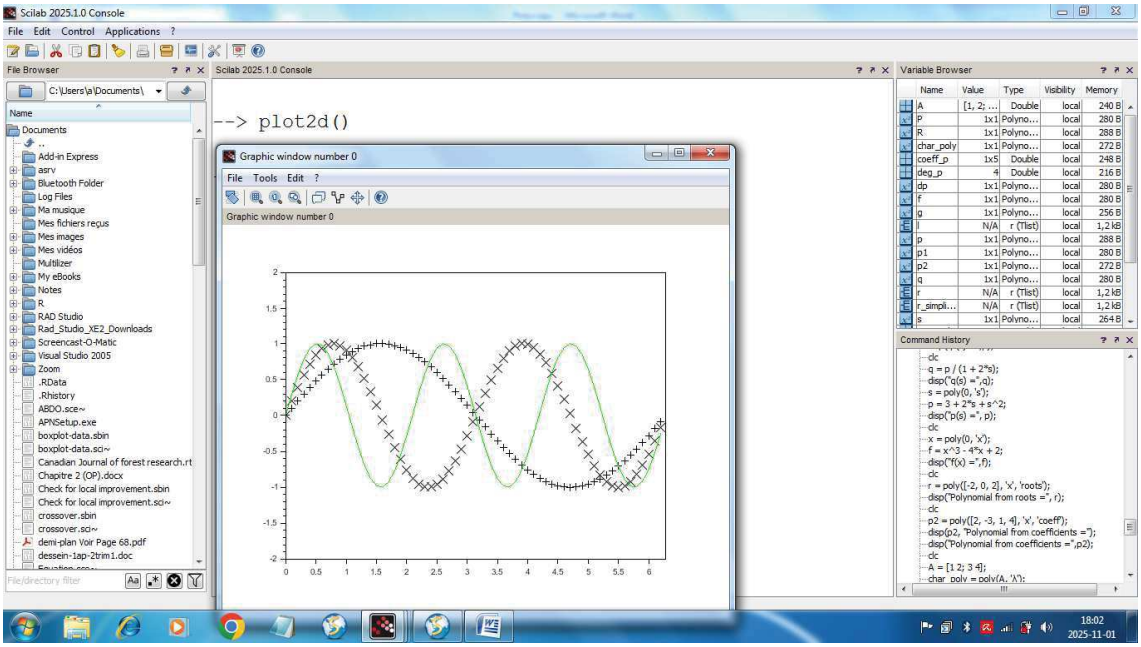
The graphic window looks like this:

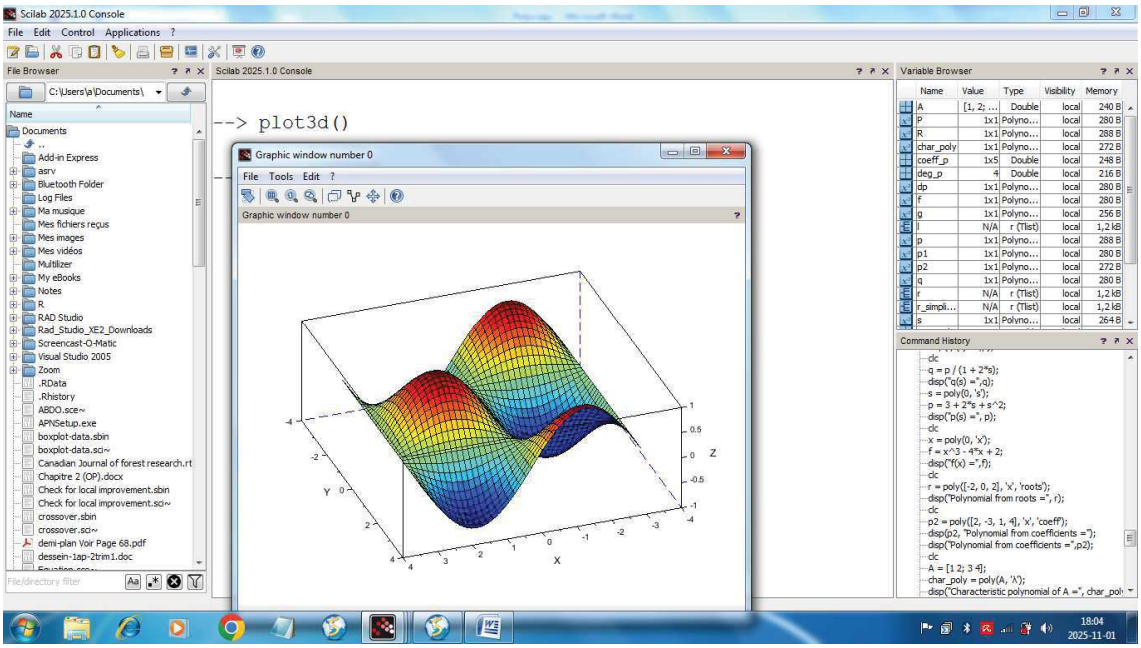


You can close the graphic window using:

```
--> close()
```

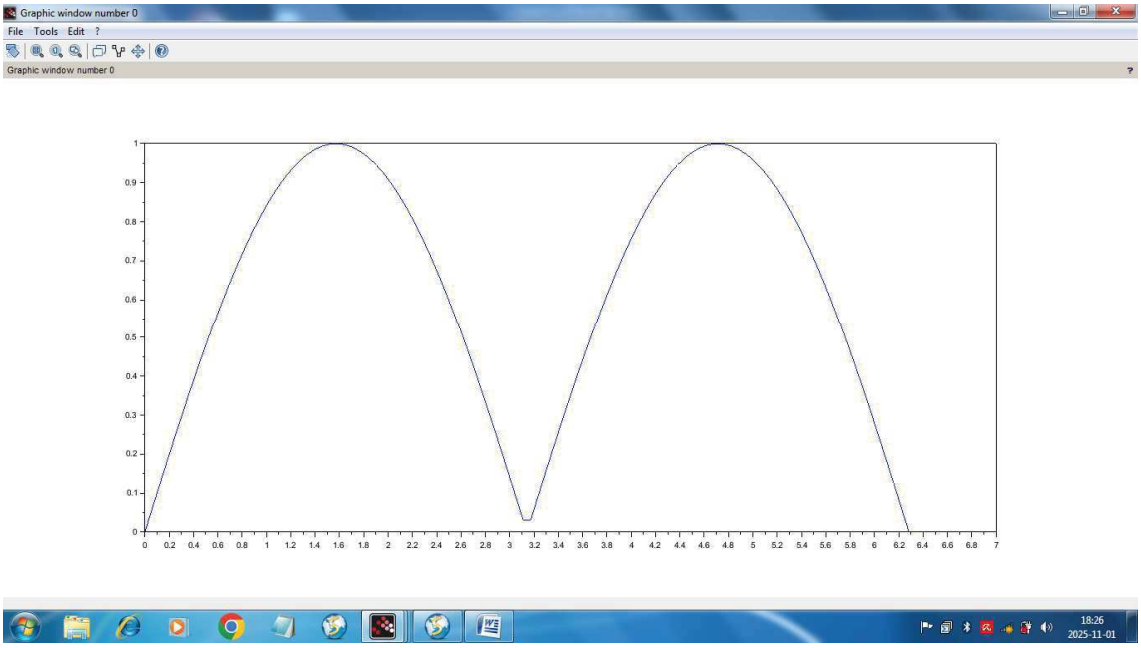
The **`scf(n)`** function, which stands for *Set Current Figure*, selects the graphics window with the number **`n`** as the current one. When you execute plotting commands such as **`plot()`** or **`plot3d()`**, Scilab sends the results to this active window. The following two graphics windows display 2D and 3D plots, respectively.



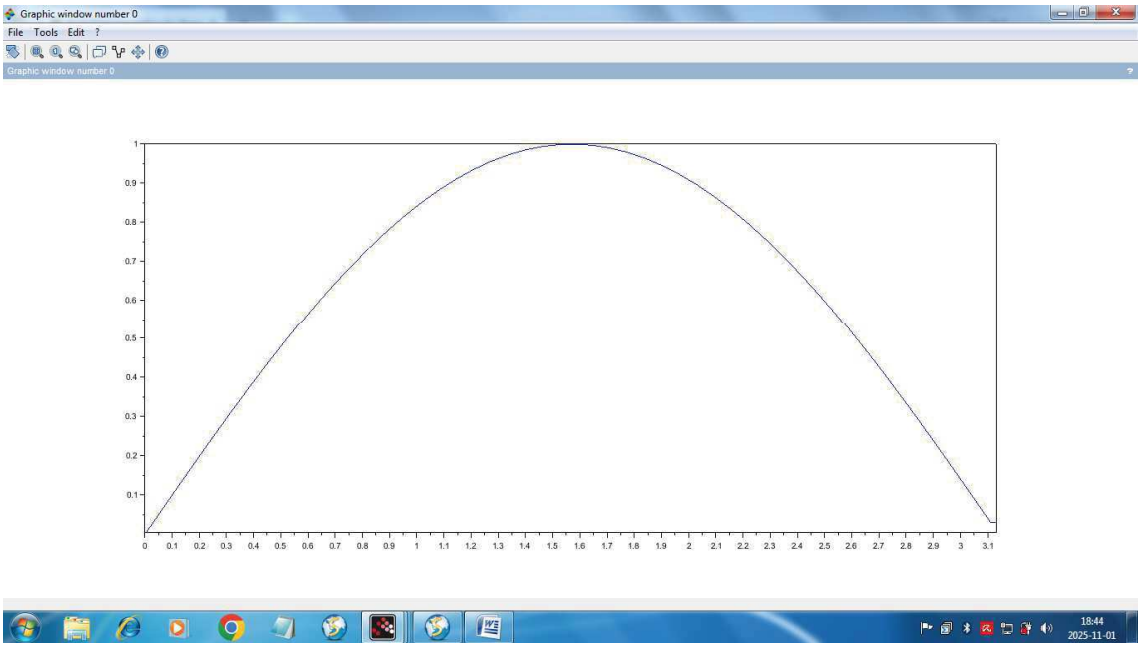


2. Graphs of functions:

```
--> function y=f(x), y=abs(sin(x)), endfunction

--> plot(linspace(0,2*%pi,100),f)
```



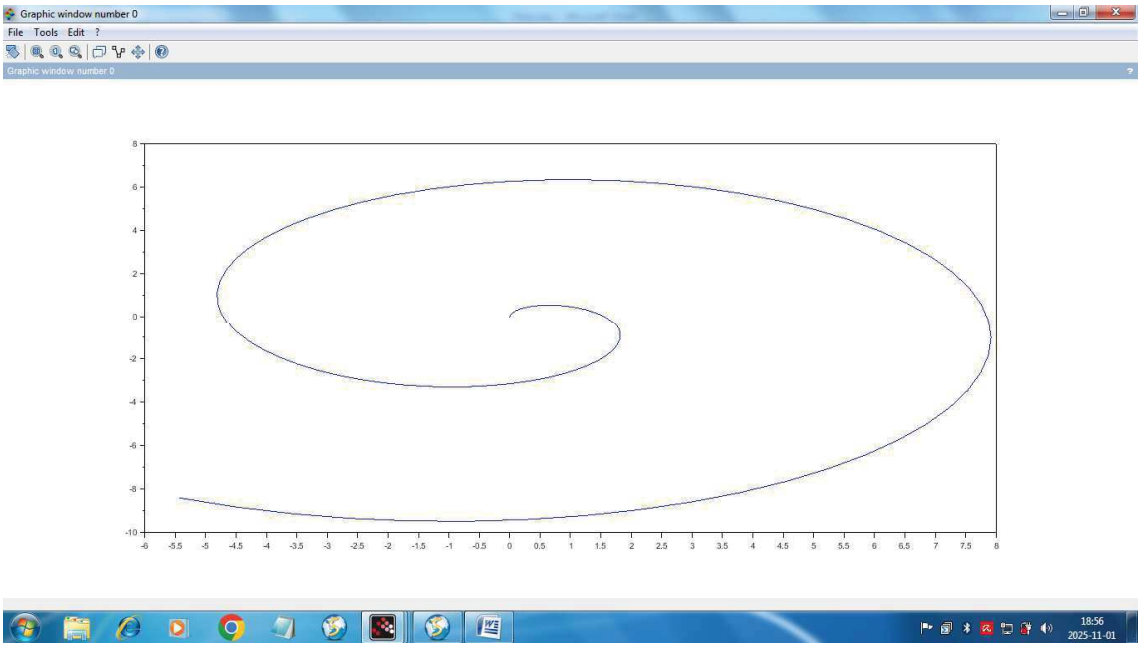
This graph and the following ones display only the contents of the drawing area. The **Tools** menu allows you to view details. Simply click on this menu, then select **Zoom** and choose a point in the window and drag the mouse to define the rectangle to zoom in on. This rectangle will then be displayed using the entire window. The figure below shows the graph obtained by zooming in on a region of the previous plot:



It is possible to repeat the operation to view more and more details. The "Original view" option in the "Tools" menu restores the original graph.

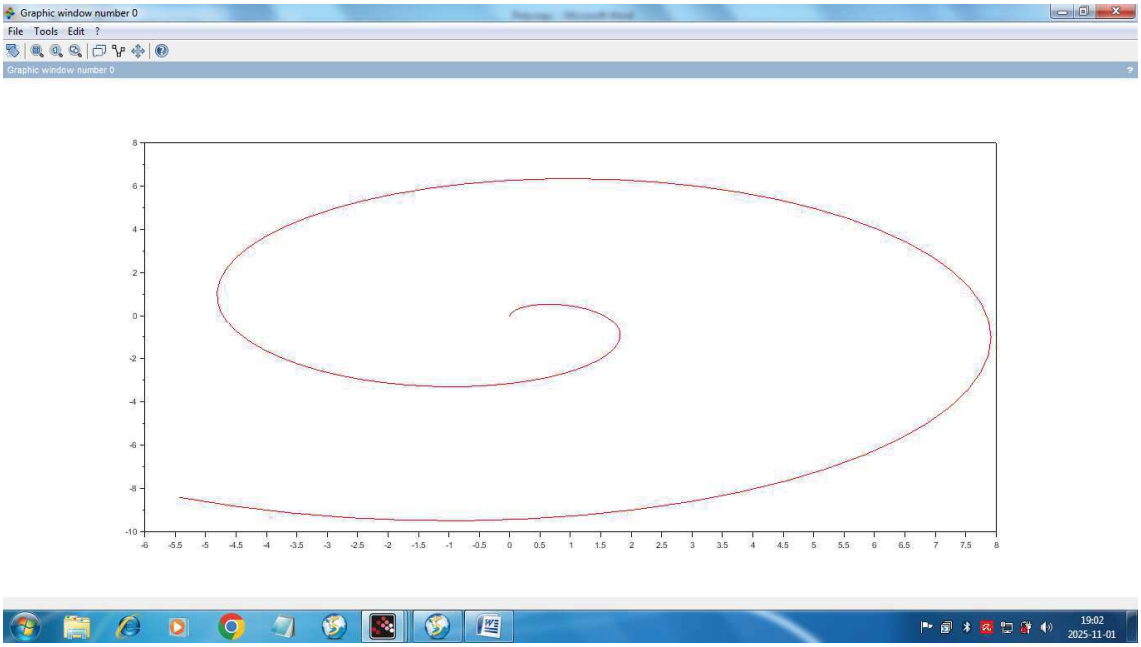
Very often, a function is represented by a set of data points **(x(i), y(i))**. In this case, the plot function can still be used.

```
--> p=linspace(0,10,100);

--> x=p.*sin(p);

--> y=p.*cos(p);

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

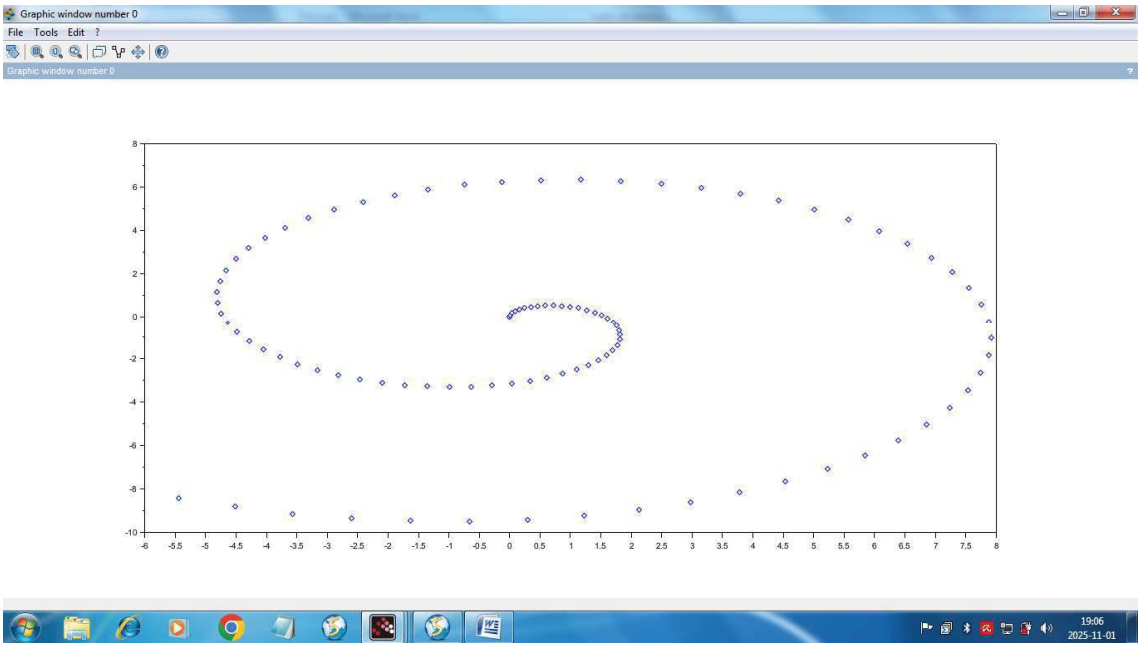


Many options can be specified through the arguments of the **`plot`** function (see the help on **`LineSpec`**).

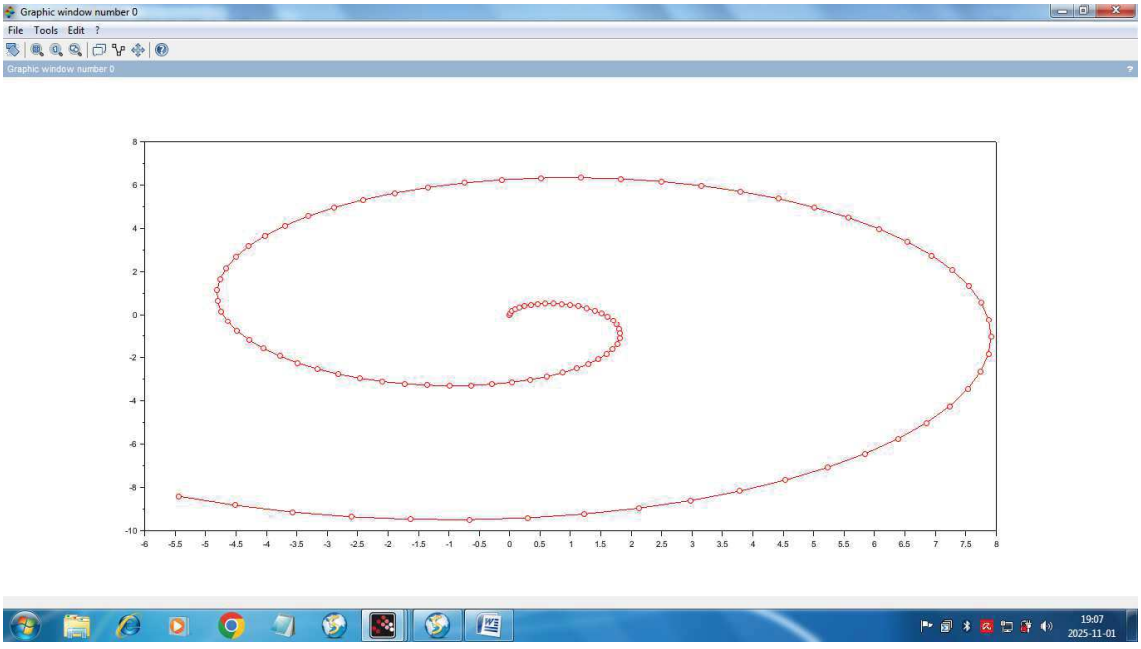
```
--> clf(), plot(x,y,'r')
```




```
--> clf(), plot(x,y,'d')
```

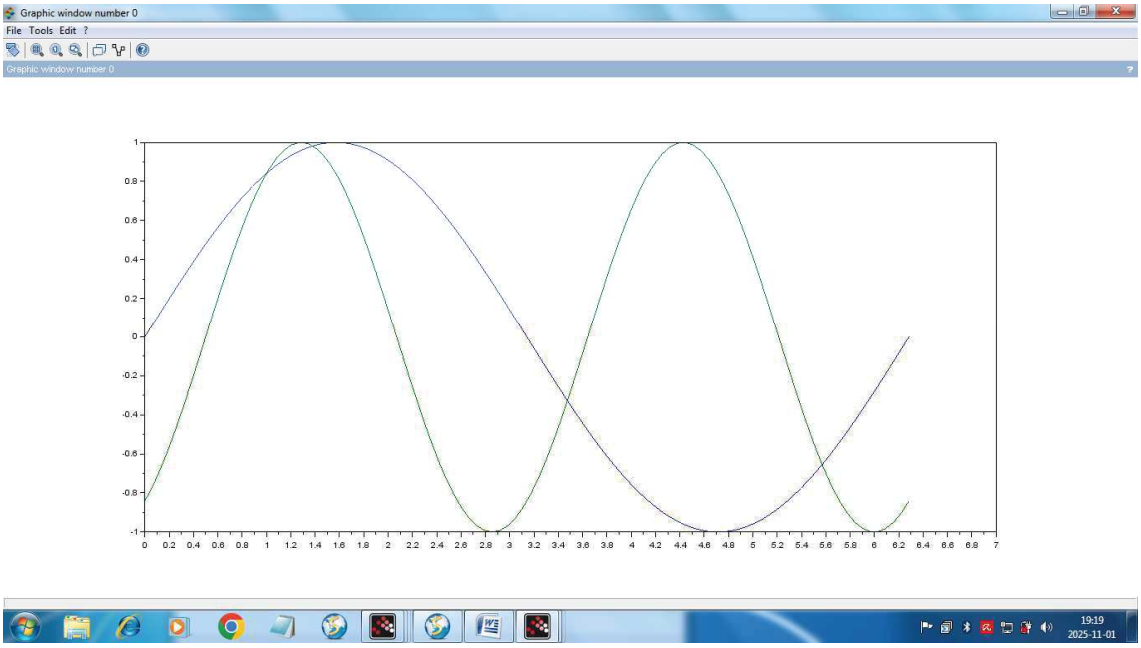



```
--> clf(), plot(x,y,'r-o')
```



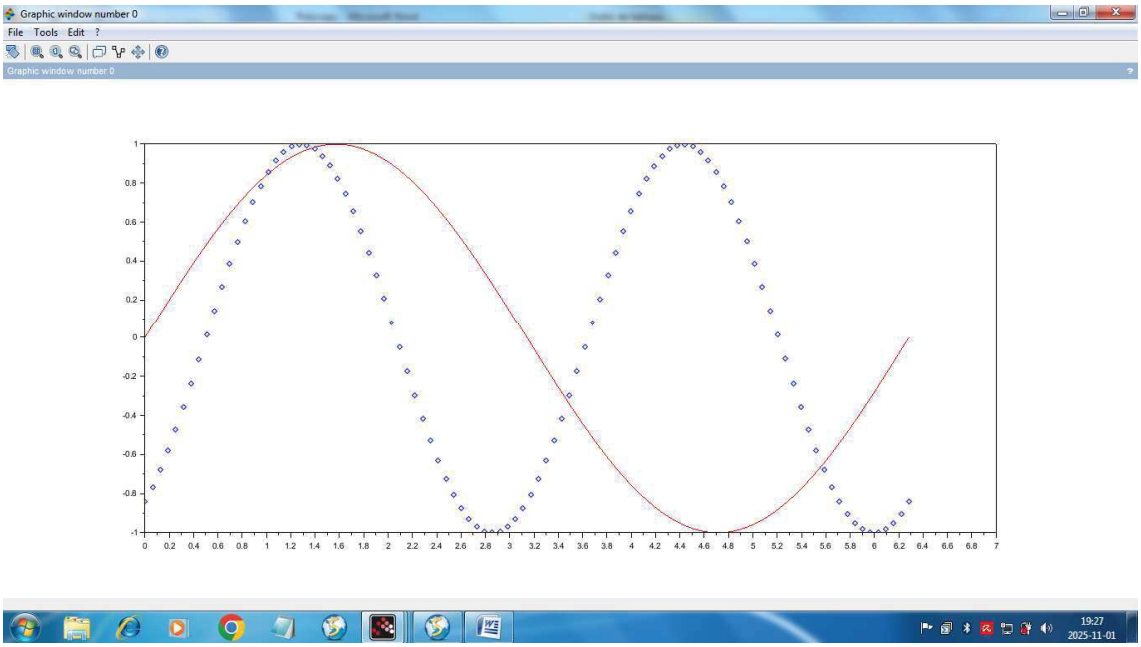
The user often wants to draw several curves in the same window to compare their behavior. Two different situations may occur: all the y-data correspond to the same discretization of the x-values.

```
--> x=linspace(0,2*%pi,100);
--> y=[sin(x)',sin(2*x-1)'];
--> clf(); plot(x,y)
```



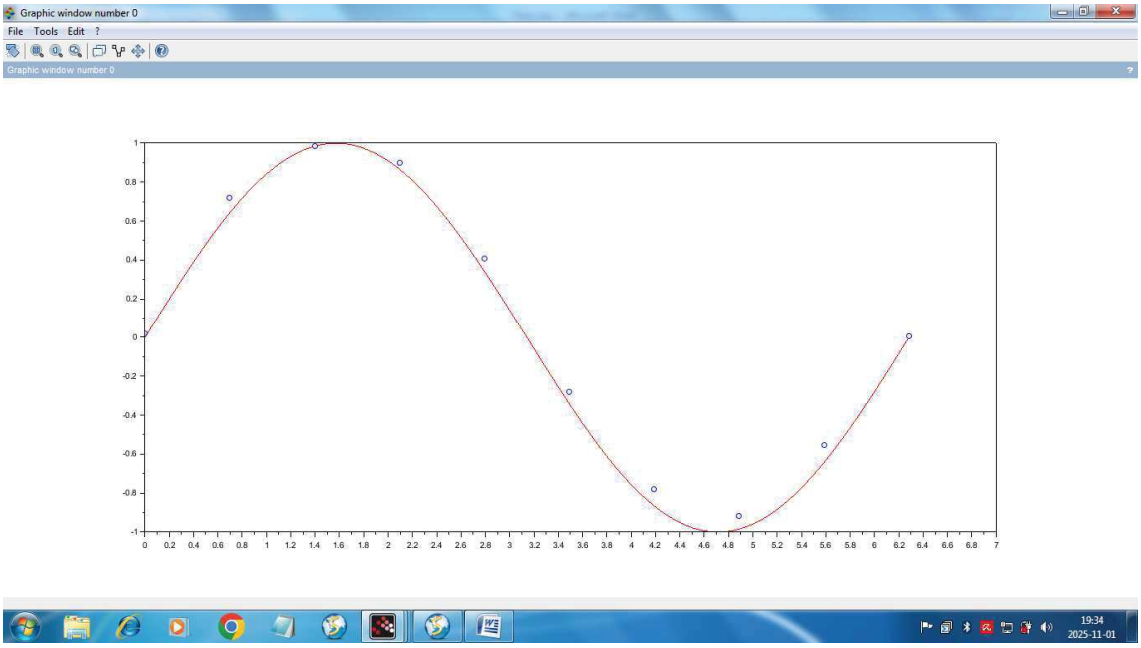
Note that each curve corresponds to a column of the matrix *y*. Each curve is drawn with a different color. It is also possible to pass several curves as arguments to the same **`plot`** function.

```
--> clf(); plot(x,sin(x),'r',x,sin(2*x-1),'d')
```



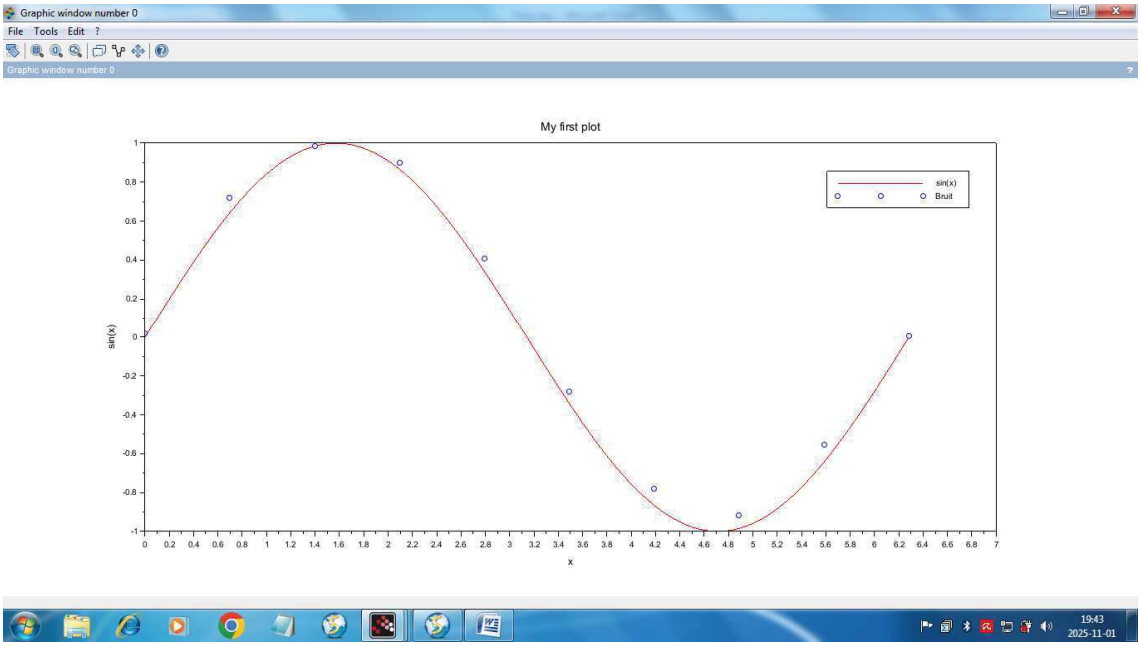
If each function has its own discretization of the x-values, the **`plot`** function can then be called several times.

```
--> x1=linspace(0,2*%pi,100);
--> x2=linspace(0,2*%pi,10);
--> clf();
--> plot(x1,sin(x1),'r')
--> plot(x2,sin(x2)+rand(x2)/10,'O')
```



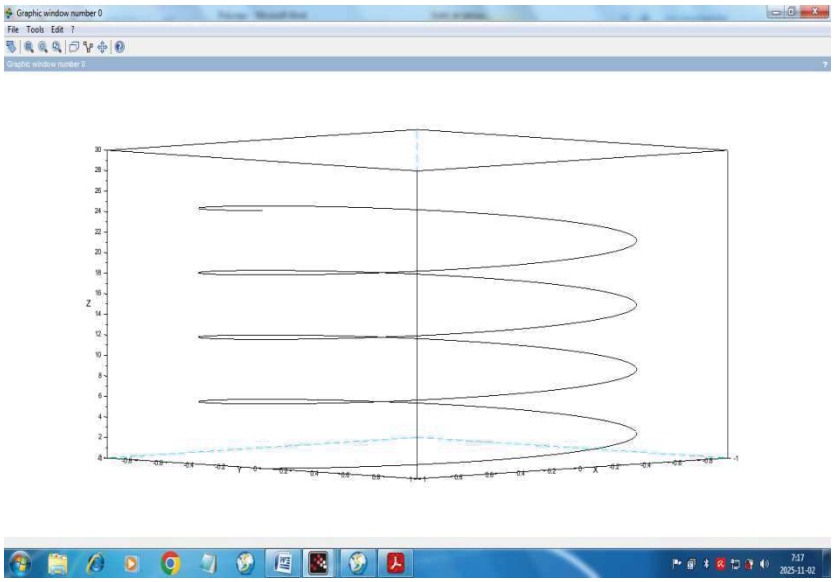
The following instructions allow you to add a legend (positioned by the user), a title, and axis labels to the graph.

```
--> legend(['sin(x)','Bruit'],5)
--> xtitle('My first plot','x','sin(x)')
```



It is, for example, possible to plot curves in 3D using the **param3d** command, such as a helix:

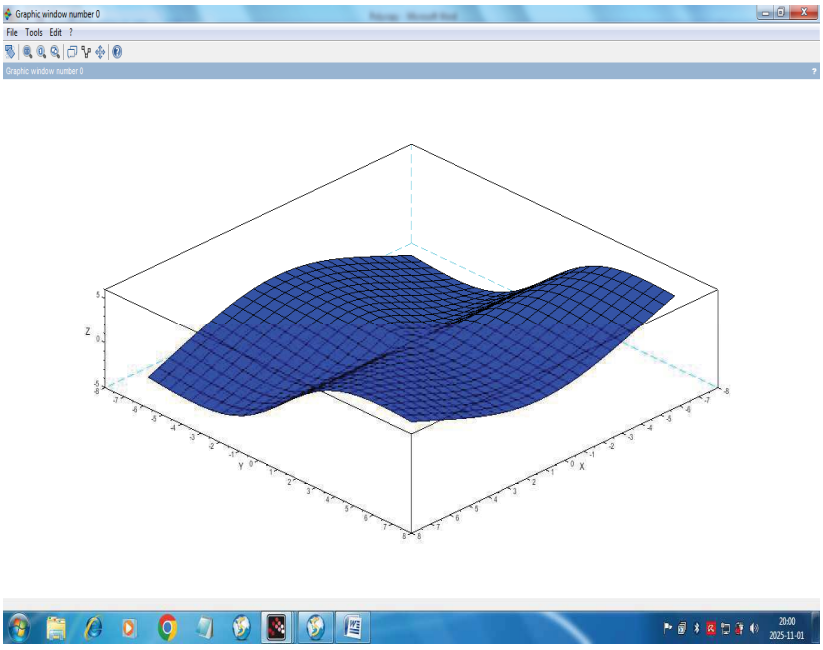
```
--> clf
--> t = 0:%pi/32:8*%pi;
--> param3d(cos(t), sin(t), t)
```



3. Analytical surfaces:

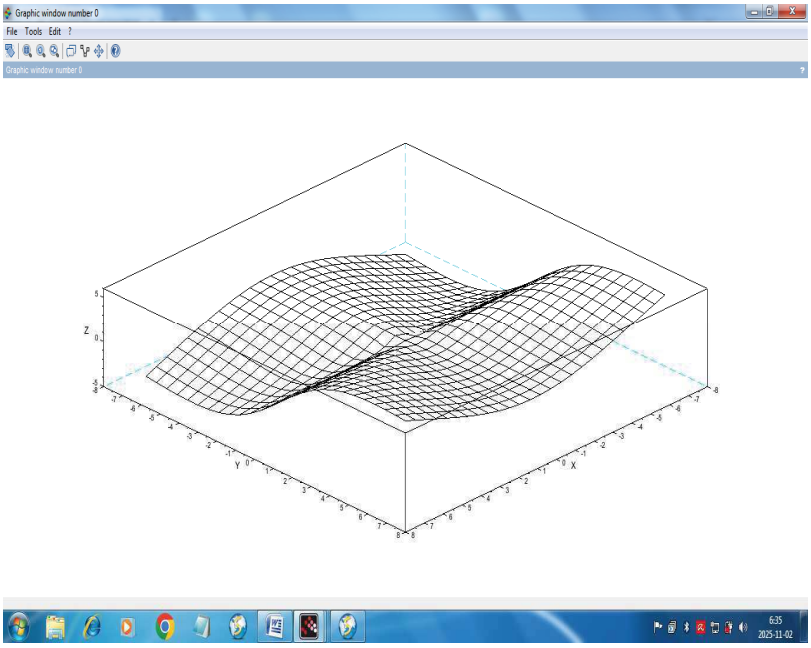
Let us first draw a surface defined by a function **z = F(x,y)**.

```
--> function z=F(x,y),...
  > z=(2*x^2*y+y^2)/(x^2+2*y^2), endfunction
--> x=-7:0.55:7;
--> clf();fplot3d(x,x,F)
```



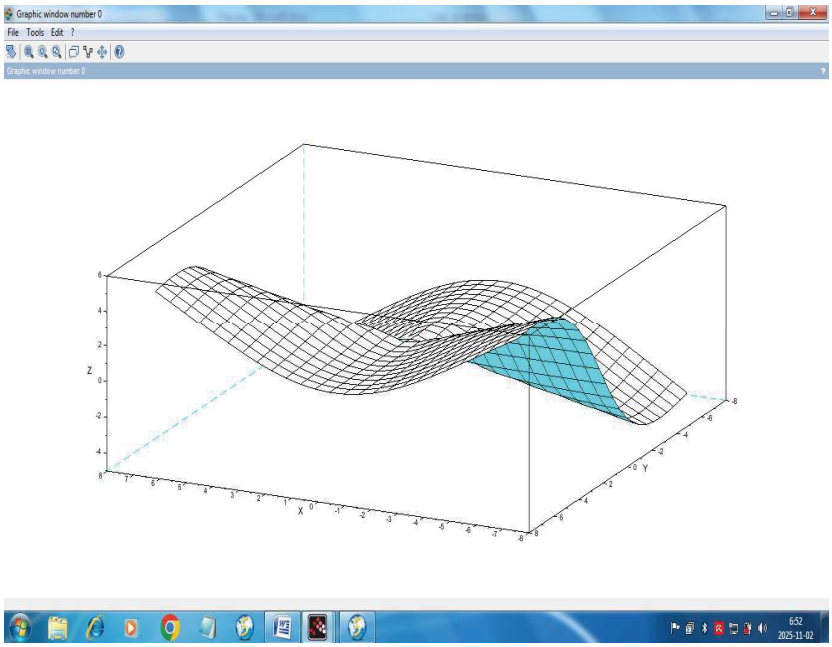
The visible surface is, by default, displayed in color number 2 from the color table. This setting can be changed through the **color_mode** property.

```
-->gce().color_mode=8
```



The **2D/3D Rotation** feature in the **Tools** menu enables adjustment of the viewing angles **θ** and **α**. The same result can be obtained by providing the optional parameters θ (theta) and α (alpha).

```
-->fplot3d(x,x,F,theta=115,alpha=63)
```



Alternatively, a surface can be represented using two vectors, **x** and **y**, specifying the discretization along the **x**- and **y**-axes, and a matrix **z**, where **z(i,j)** denotes the elevation at the point (**x(i), y(j)**). The surface corresponding to the function **F** below can then be constructed by computing the matrix **z** as follows:

```
--> z=feval(x,x,F);
--> plot3d(x,x,z)
```

The surface, represented in this manner, can be visualized using the **surf** function.

```
--> surf(x,x,z)
```