Matlab Laboratory 5. 2-D and3-D plots in Matlab

Aim of the laboratory

- 2D and 3D graphical representations
- Customization of Matlab plots

Necessary equipment

- Workstations with MATLAB

Theoretical Approach

Graphical representations in Matlab are a very useful tool for function visualization. A graph, or a plot, can provide valuable information for function analysis and characterization. For a proper graphical representation, one must follow some basic steps:

- Definition of the function data,
- Definition of the plot interface,
- Plotting the function,
- Editing the plot properties.

These steps will first be detailed on 2-dimensional plots, and then they will be illustrated for 3-dimensional plots as well.

2-dimensional plots

A 2-dimensional plot prints the graphical representation of a one-variable function *y*, defined as

$$y = f(x)$$

Graphical representation is performed in a 2-dimensional coordinate system, where the function variable *x* is placed along the horizontal axis and the function values *y* are placed along the vertical axis. The graphical representation is exemplified using the ***plot*** function:

```
plot(x,y);
```

where both arguments of the ***plot*** function are vectors which must have the same length.

Definition of the function data

Definition of the function data for a 2-D plot requires:

- definition of the variable variation domain,
- expression of the function along the variable variation domain.

Consider for example the definition of a sine wave:

```
x = -pi:pi/100:pi;
y_sin = sin(x);
y_cos = cos(x);
```

The sine wave is defined on a 2π signal period. Accordingly, the variable was normalized to the $-\pi$ to π variation domain with a $\pi/100$ step size. Another example is the definition of a square wave with an arbitrary duty cycle, on the same variable variation domain *x* defined beforehand:

```
duty = 30;
y_sq = square(x,duty);
```

as well as a triangular wave with an arbitrary skew:

```
skew = 0.5;
y_tr = sawtooth(x,skew);
```

Further signal waveforms which can be defined in Matlab are **chirp**, **cos**, **diric**, **gauspuls**, **pulstran**, **rectpuls**, **sinc** and **tripuls**, which can be consulted in the Matlab help.

In order to have an arbitrary function definition domain, e.g. time domain, one should consider the relationship between frequency and time as follows

```
f = 50;                 % frequency
per = 1/f;              % period
t = 0:per/100:4*per;    % time domain
x = 2 * pi * f * t;     % variable definition domain
```

This code sequence defines a 4 period variable variation domain with a 100 point per period step size. To be noticed is that the 100 points per period step size is a useful compromise between resolution and data size. If a function requires a smaller resolution, a smaller step size can be chosen. This however results in larger data structures.

Another issue to be considered when defining the time domain is having multiple waveforms with different frequencies defined over the same variable variation domain. Regardless of the number of signals to be defined, the time domain should be the same and should be defined for the smallest frequency, i.e. the largest period. Accordingly, the higher frequency signals will exhibit a higher number of oscillations in the same time interval:

```
f1 = 50;                % frequency for sin1
f2 = 200                % frequency for sin2
per = 1/f1;             % period
t = 0:per/100:4*per;    % time domain
x1 = 2 * pi * f1 * t;   % variable definition domain for sin1
x2 = 2 * pi * f2 * t;   % variable definition domain for sin2, same t as
                        % for sin1
```

Definition of the plot interface

A new plot window is created using the ***figure*** function. Consider for example plotting the sine wave defined in the previous section.

```
figure();
plot(x,y_sin);
```

The function plot resulting after running this Matlab section is illustrated in figure 1.

A name can be added to the plot window using:

```
figure('Name','Sine Waves','NumberTitle','off');
```

For further properties of the ***figure*** function consult the Matlab help browser.

If multiple plot windows are required, it is useful to consider numbering the windows. Then, one may also switch in-between already opened windows:

```
figure(1);         % creates new plot window numbered 'Figure 1'
plot(x,y_sin);
figure(2);         % creates new plot window numbered 'Figure 2'
plot(x,y_sq);
figure(3);         % creates new plot window numbered 'Figure 3'
plot(x,y_tr);
figure(1);         % switches back to plot window numbered 'Figure 1'
```

Plotting a new function into an existing window will overwrite the former plot. In order to keep the former plot in the same window, one is required to activate the ***hold*** function as follows

```
figure();        % creates new plot window numbered 'Figure 1'
plot(x,y_sin);
hold on;
plot(x,y_cos);
```

The result is illustrated in figure 2.

To deactivate the ***hold*** function within a plot window, which will cause overwriting all plots from that window at the next ***plot*** instruction, one must type

```
hold off;
```

The plot window may further be divided into subplots using the ***subplot*** function as follows. The syntax of the ***subplot*** function is

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

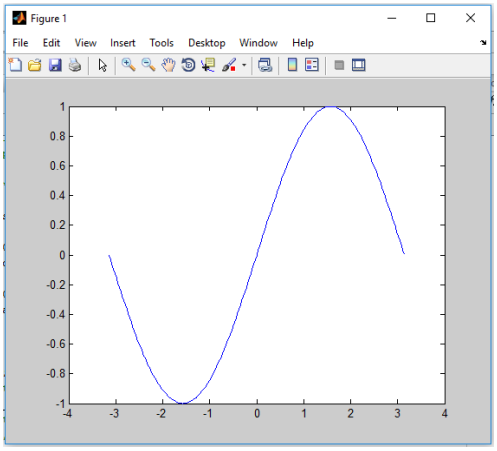

Figure 1

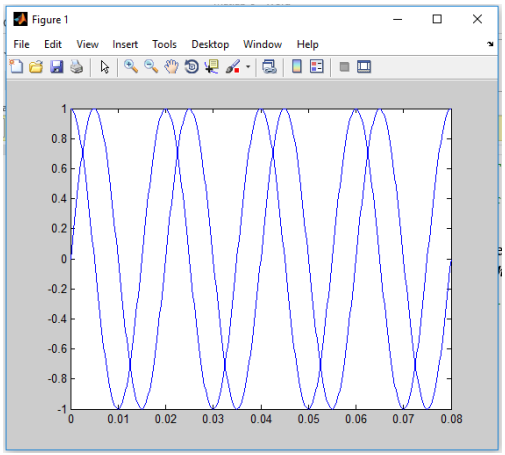

Figure 2

The ***subplot*** function performs the division of the plotting window into *m* x *n* sub-windows, and activates the p^{th} sub-window, counted from left to right and from top to bottom. Consider for exemplification

```
figure();
subplot(2,2,1);
plot(t,y_sin);
subplot(2,2,2);
plot(t,y_cos);
subplot(2,2,3);
plot(x,y_sq);
subplot(2,2,4);
plot(x,y_tr);
```

with the result illustrated in figure 3.

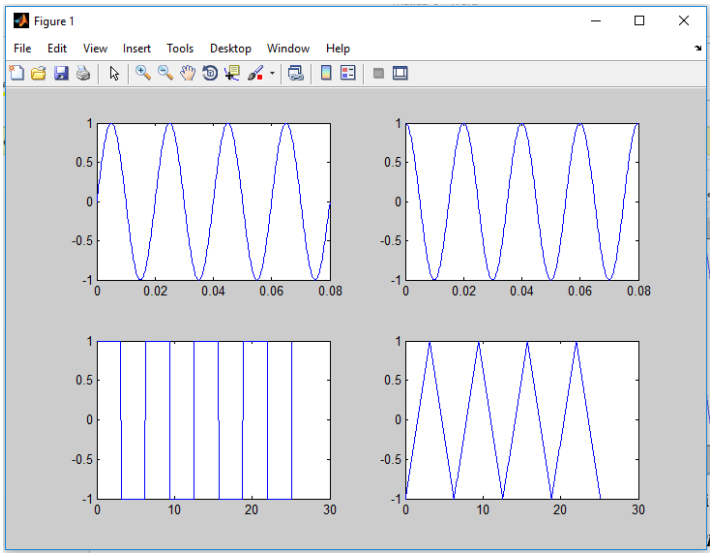

Figure 3

Plotting the function

The ***plot*** function is indeed the most straightforward way to plot the shape of a function in Matlab:

```
plot(x,y);
```

The ***plot*** function basically connects the $x(i)$ and $y(i)$ coordinate points, with i ranging from 1 to ***length***(y). Accordingly, the function plot illustrates the shape of the function y vs. the function definition domain x.

To be noticed is that if the x argument is missing, the function ***plot*** will illustrate the function y vs. a linear space ranging from 1 to length(y):

```
plot(y);
```

Then again, the ***plot*** function takes pairs of vectors x_k and y_k, k=1…n, where y_k is the function to be plotted and x_k is the corresponding function definition domain.

```
plot(x1,y1,x2,y2,x3,y3);
```

In this case, the n functions are plotted simultaneously in the same figure. The same result would have been achieved with turning on the ***hold*** function for the individual plots.

Editing the plot properties

Editing the plot properties basically refers to editing all the elements of the function plot:

- editing the plot line,
- naming the plot
- labeling the axes,
- placing grids,
- printing text on the plot,
- etc.

For plot editing, the syntax of the ***plot*** function changes to

```
plot(x,y,S);
```

where S is the property specifier, and defines the specifications regarding line style, color and markers.

The line style specifiers are listed in Table 1.

Table 1

Specifier	Line Style
‘–’	Solid line (default)
‘– –’	Dashed line
‘:’	Dotted line
‘-.’	Dash-dotted line
‘none’	No line

The color specifiers are listed in Table 2.

Table 2

Specifier	Color
‘r’	Red
‘g’	Green
‘b’	Blue
‘c’	Cyan
‘m’	Magenta
‘y’	Yellow
‘k’	Black
‘w’	While

The marker specifiers are listed in Table 3.

Table 3

Specifier	Marker
‘+’	Plus sign
‘o’	Circle
‘*’	Asterisk
‘.’	Point
‘x’	Cross
'square' or 's'	Square
'diamond' or 'd'	Diamond
‘^’	Upward-pointing triangle
‘v’	Downward-pointing triangle
‘>’	Right-pointing triangle
‘<’	Left-pointing triangle
‘pentagram’ or ‘p’	Five-pointed star (pentagram)
‘hexagram’ or ‘h"’	Six-pointed star (hexagram)
‘none’	No marker (default)

The S specifier is then a string consisting of the individual specifiers listed in Tables 1, 2 and 3. For exemplification, consider plotting one period of a sine and cosine wave in the same figure using different colors:

```
x = -pi:0.1:pi;
y_sin = sin(x);
y_cos = cos(x);

Fig1=figure('Name','Sine Waves','NumberTitle','off');
plot(x,y_sin,'r',x,y_cos,'b');
```

Further on, consider for exemplification changing the line style as follows.

```
plot(x,y_sin,':r',x,y_cos,'--b');
```

Markers can also be added to the function plot considering for example

```
plot(x,y_sin,':r+',x,y_cos,'--bo');
```

Additionally, plot legend can be added with typing

```
legend('sine','cosine');
```

The result is illustrated in figure 4

Further on, the ***plot*** function allows to change the line width, the marker size, the marker color and the marker edge color. For this purpose, one must use pairs of specifier and values as follows [Matlab help]:

- LineWidth – specifies the width in points of the line,
- MarkerEdgeColor – specifies the color of the marker or the edge color for filled markers,
- MarkerFaceColor – specifies the color of the filled markers,
- MarkerSize – specifies the size of the marker in points.

Consider for example

```
plot(x,y_sin,'-g^',...
                'LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','y',...
                'MarkerSize',5)
```

In this case, the sine plot as illustrated in figure 5.

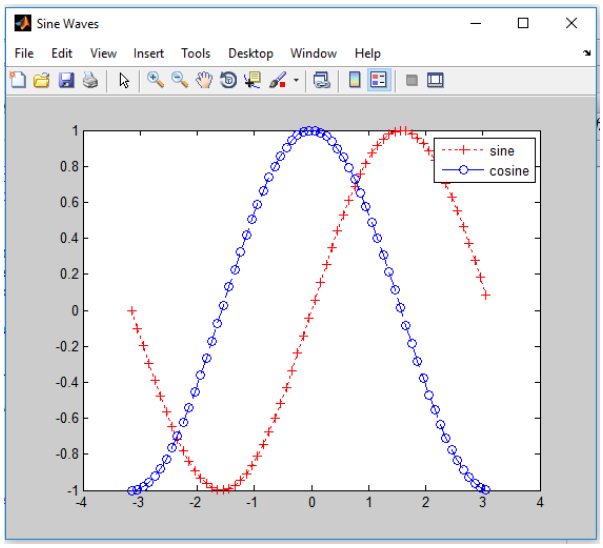

Figure 4.

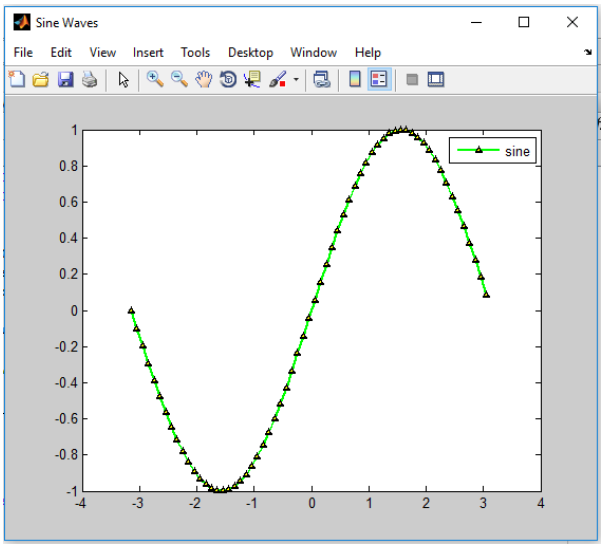

Figure 5.

A title can be added to the plot using the ***title*** function as follows:

```
title('Plot Title');
```

Labeling the plot axes is performed with the ***xlabel*** and ***ylabel*** functions which take strings as arguments as follows:

```
xlabel('x');
ylabel('sin(x)');
```

Scaling the axes is performed with the ***axis*** function which takes as arguments the minimum and maximum values for the variation ranges along the *x* and *y* axes respectively. For further properties of the ***axis*** function consult the Matlab help.

```
axis([xmin xmax ymin ymax])
```

Printing a grid o the figure is done with the ***grid*** function which must be activated as follows:

```
grid on;
```

The grid can later be deactivated by turning it off using

```
grid off;
```

Consider for example

```
title('One Period of a Sine Wave');
xlabel('x');
```

```
ylabel('sin(x)');
axis([-5,5,-1.5,1.5]);
grid on;
```

which changes the sine plot as illustrated in figure 6.

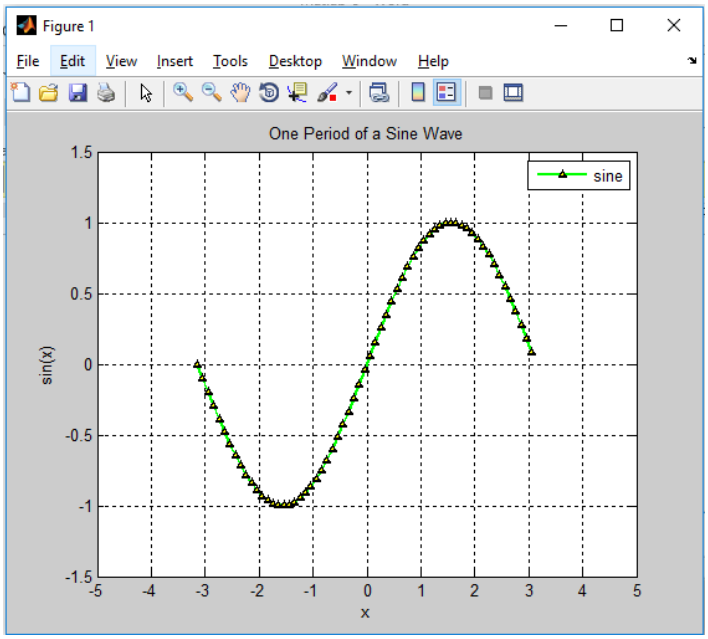

Figure 6

The *ezplot* function

A direct way to plot the graph of a function over a given variation domain is the ***ezplot*** function. The syntax of the ***ezplot*** function is

```
ezplot(fun,[min,max]);
```

where *fun* is the function expression given as a string, and [*min*, *max*] is the variation domain of the function variable. If the variation domain is missing from the function call, Matlab considers the implicit [-2π, 2π] variation range.

Consider for exemplification plotting a sine wave over a 2π period

```
ezplot('sin(x)',[-pi,pi]);
```

which results in the plot from figure 7. To be noticed is that, in contrast to the ***plot*** function where the plot title and the axis labels must be specified explicitly, the ***ezplot*** function labels the plot automatically.

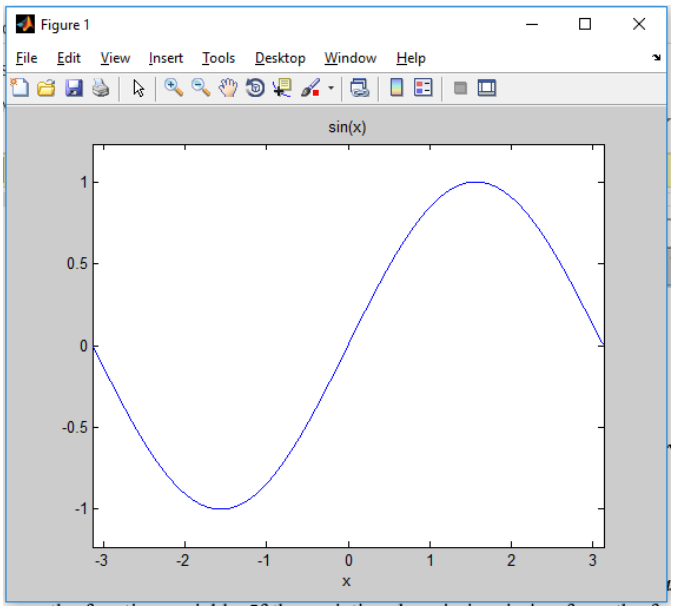

Figure 7

Additionally, the ***ezplot*** function works together with the ***grid*** and ***hold*** functions to turn the grid on and to plot multiple graphs in the same window.

3-dimensional plots in Matlab

The ***plot3*** function is the 3-dimensional alternative to the ***plot*** function. The syntax of the ***plot3*** function is

```
plot3(x,y,z);
```

which interconnects the *x*(*i*), *y*(*i*) and *z(i)* coordinate points. Accordingly, the result of the ***plot3*** function is a 3-D curve, as illustrated in the example below

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
xlabel('sin(t)')
ylabel('cos(t)')
zlabel('t')
grid on
```

which results in the plot from figure 8.

Editing the 3-dimensional plot is done using a property specifier

```
plot3(x,y,z,S);
```

where S defines the specifications regarding line style, color and markers as listed in Tables 1, 2 and 3 respectively.

Additionally, the ***plot3*** function further allows adding plot title, axis labels, turning on the grid and plotting multiple curves in the same window, as was the case for the ***plot*** function.

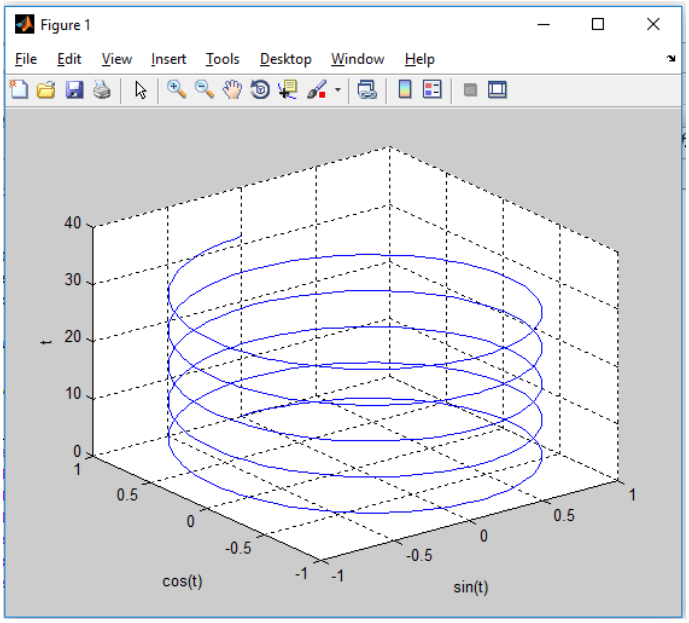

Figure 8

To be noticed is that, rather than plotting a 2-variable function *f*(*x*,*y*), the ***plot3*** function draws a curve in the 3-dimensional space which consists in interconnecting successive 3-coordinate points. In order to plot 2-variable functions, one should use the ***mesh*** and ***surf*** functions as follows. For 2-variable function plots however, the definition of the variable variation domain must be done accordingly. The ***meshgrid*** function will be used for this purpose.

The *meshgrid* function

The ***meshgrid*** function performs a conversion of the domain specified by two vectors *x* and *y* into matrices *X* and *Y*. The lines of the resulting *X* matrix will be copies of the *x* vector and the columns of the resulting *Y* matrix will be copies of the *y* vector. Consider for exemplification

```
[X, Y] = meshgrid(1:3)
```

which returns

```
X =

     1     2     3
     1     2     3
     1     2     3

Y =

     1     1     1
     2     2     2
     3     3     3
```

For different *X* and *Y* matrices consider

```
[X, Y] = meshgrid(1:3,4:6)
```

which returns

```
X =

     1     2     3
     1     2     3
     1     2     3


Y =

     4     4     4
     5     5     5
     6     6     6
```

These matrices are then used to create a 3-D plot of the two-variable function.

The *mesh* function

The ***mesh*** function is used to plot a 2-variable function in the shape of a wireframe surface. Consider for exemplification the function defined as

$$f(x,y) = \sin(x) \cdot \sin(y), x, y \in (-\pi, \pi]$$

The Matlab code which creates the mesh of the function is listed as follows.

```
[x, y] = meshgrid(-pi:pi/10:pi);
z = sin(x) .* sin(y);
mesh(z);
```

The resulting mesh is illustrated in figure 9.

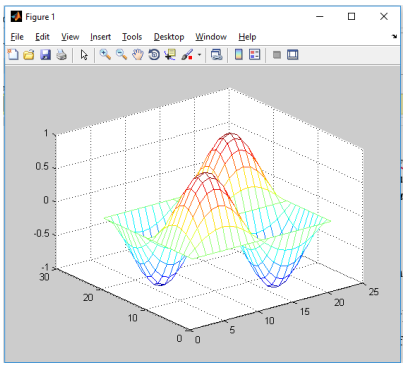

Figure 9

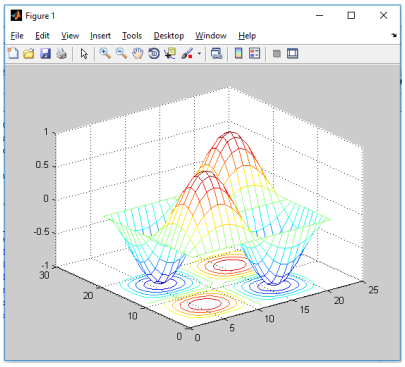

Figure 10

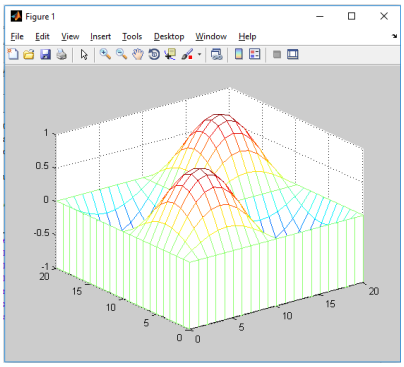

Figure 11

The mesh can further be edited by adding a contour plot using the ***meshc*** function:

```
meshc(z);
```

resulting in the mesh illustrated in figure 10, or creating a waterfall plot using the ***meshz*** function:

```
meshz(z);
```

resulting in the mesh illustrated in figure 11.

Additionally, the ***mesh*** function allows adding plot title and axis labels, editing the axes, turning on the grid and plotting multiple curves in the same window.

The *surf* function

The ***surf*** function is used to plot a 2-variable function in the shape of a rectangular surface. The surface of the same function as in the previous example is drawn with the Matlab code listed as follows.

```
[x, y] = meshgrid(-pi:pi/10:pi);
z = sin(x) .* sin(y);
surf(z);
```

The resulting surface is illustrated in figure 12.

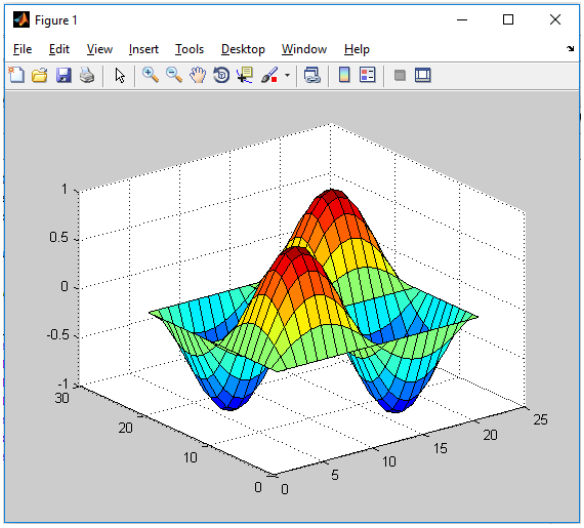

Figure 12

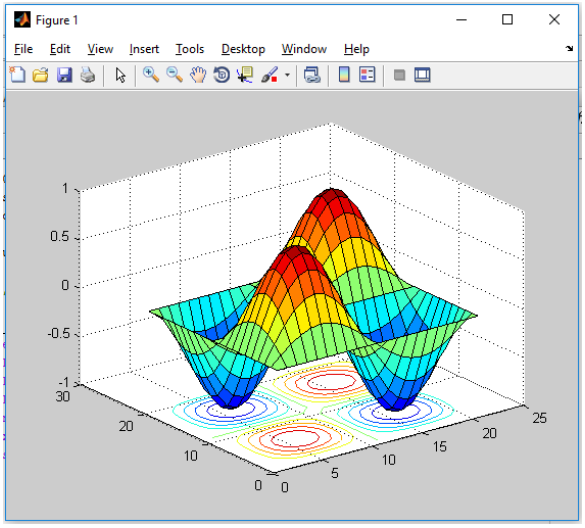

Figure 13

The surface can further be edited by adding a contour plot using the ***surfc*** function:

```
surfc(z);
```

resulting in the surface illustrated in figure 13. Additionally, the ***surf*** function allows adding plot title and axis labels, editing the axes, turning on the grid and plotting multiple curves in the same window.