- The line specifiers, properties, and property values are the same as in 2-D plots (see Section 5.1).

For example, if the coordinates x, y, and z are given as a function of the parameter t by

$$x = \sqrt{t}\sin(2t)$$
$$y = \sqrt{t}\cos(2t)$$
$$z = 0.5t$$

a plot of the points for $0 \le t \le 6\pi$ can be produced by the following script file:

```
t=0:0.1:6*pi;
x=sqrt(t).*sin(2*t);
y=sqrt(t).*cos(2*t);
z=0.5*t;
plot3(x,y,z,'k','linewidth',1)
grid on
xlabel('x'); ylabel('y'); zlabel('z')
```

The plot shown in Figure 10-1 is created when the script is executed.

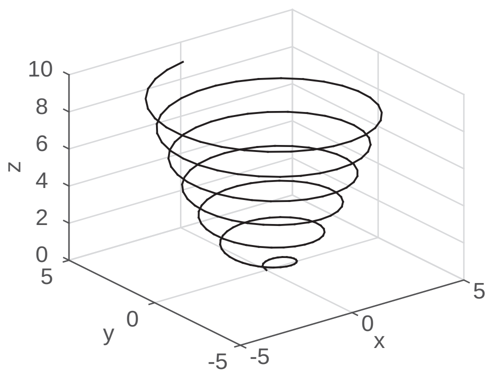


Figure 10-1: A plot of the function $x = \sqrt{t}\sin(2t)$, $y = \sqrt{t}\cos(2t)$, $z = 0.5t$ for $0 \le t \le 6\pi$.

10.2 *MESH AND SURFACE PLOTS*

Mesh and surface plots are three-dimensional plots used for plotting functions of the form $z = f(x, y)$ where x and y are the independent variables and z is the dependent variable. It means that within a given domain the value of z can be calculated for any combination of x and y. Mesh and surface plots are created in three steps. The first step is to create a grid in the x y plane that covers the domain of the function. The second step is to calculate the value of z at each point of the grid. The third step is to create the plot. The three steps are explained next.

Creating a grid in the *x y* plane (Cartesian coordinates):

The grid is a set of points in the *x y* plane in the domain of the function. The density of the grid (number of points used to define the domain) is defined by the user. Figure 10-2 shows a grid in the domain $-1 \le x \le 3$ and $1 \le y \le 4$. In this

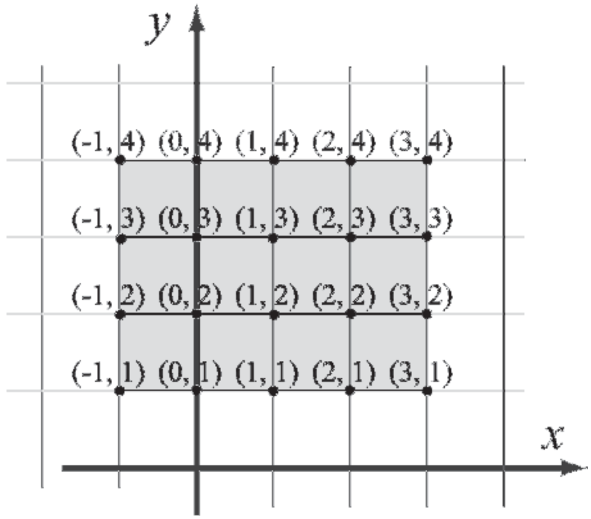


Figure 10-2: A grid in the *x y* plane for the domain $-1 \le x \le 3$ and $1 \le y \le 4$ with spacing of 1.

grid the distance between the points is one unit. The points of the grid can be defined by two matrices, *X* and *Y*. Matrix *X* has the *x* coordinates of all the points, and matrix *Y* has the *y* coordinates of all the points:

$$X = \begin{bmatrix} -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 4 \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} 4 & 4 & 4 & 4 & 4 \\ 3 & 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The *X* matrix is made of identical rows since in each row of the grid the points have the same *x* coordinate. In the same way the *Y* matrix is made of identical columns since in each column of the grid the *y* coordinate of the points is the same.

MATLAB has a built-in function, called `meshgrid`, that can be used for creating the X and Y matrices. The form of the `meshgrid` function is:

```
[X,Y] = meshgrid(x,y)
```

`X` is the matrix of the x coordinates of the grid points.
`Y` is the matrix of the y coordinates of the grid points.

`x` is a vector that divides the domain of x.
`y` is a vector that divides the domain of y.

In the vectors `x` and `y` the first and last elements are the respective boundaries of the domain. The density of the grid is determined by the number of elements in the vectors. For example, the mesh matrices X and Y that correspond to the grid in Figure 10-2 can be created with the `meshgrid` command by:

```
>> x=-1:3;
>> y=1:4;
>> [X,Y]=meshgrid(x,y)
X =
    -1     0     1     2     3
    -1     0     1     2     3
    -1     0     1     2     3
    -1     0     1     2     3
Y =
     1     1     1     1     1
     2     2     2     2     2
     3     3     3     3     3
     4     4     4     4     4
```

Once the grid matrices exist, they can be used for calculating the value of z at each grid point.

Calculating the value of z at each point of the grid:

The value of z at each point is calculated by using element-by-element calculations in the same way it is used with vectors. When the independent variables x and y are matrices (they must be of the same size), the calculated dependent variable is also a matrix of the same size. The value of z at each address is calculated from the corresponding values of x and y. For example, if z is given by

$$z = \frac{xy^2}{x^2 + y^2}$$

the value of z at each point of the grid above is calculated by:

```
>> Z = X.*Y.^2./(X.^2 + Y.^2)
```

```
Z =
   -0.5000         0    0.5000    0.4000    0.3000
   -0.8000         0    0.8000    1.0000    0.9231
   -0.9000         0    0.9000    1.3846    1.5000
   -0.9412         0    0.9412    1.6000    1.9200
```

Once the three matrices have been created, they can be used to plot mesh or surface plots.

Making mesh and surface plots:

A mesh or surface plot is created with the `mesh` or `surf` command, which has the form:

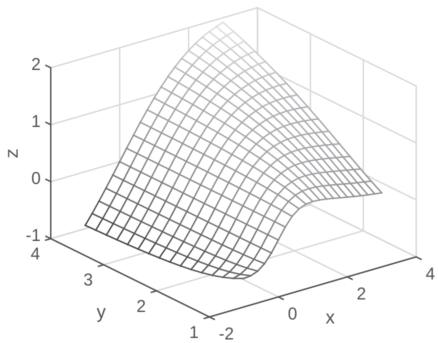
`mesh(X,Y,Z)`   `surf(X,Y,Z)`

where `X` and `Y` are matrices with the coordinates of the grid and `Z` is a matrix with the value of z at the grid points. The mesh plot is made of lines that connect the points. In the surface plot, areas within the mesh lines are colored.

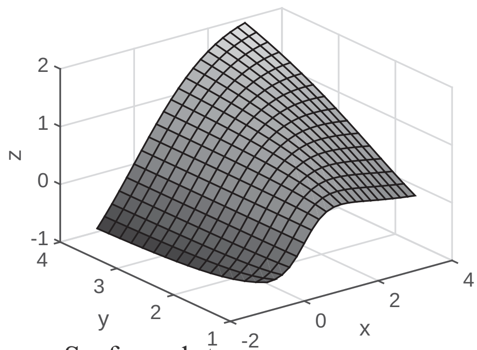
As an example, the following script file contains a complete program that creates the grid and then makes a mesh (or surface) plot of the function $z = \frac{xy^2}{x^2+y^2}$ over the domain $-1 \le x \le 3$ and $1 \le y \le 4$.

```
x=-1:0.1:3;
y=1:0.1:4;
[X,Y]=meshgrid(x,y);
Z=X.*Y.^2./(X.^2+Y.^2);
mesh(X,Y,Z)                    Type surf(X,Y,Z) for surface plot.
xlabel('x'); ylabel('y'); zlabel('z')
```

Note that in the program above the vectors `x` and `y` have a much smaller spacing than the spacing earlier in the section. The smaller spacing creates a denser grid. The figures created by the program are:



Mesh plot



Surface plot

Additional comments on the mesh command:

- The plots that are created have colors that vary according to the magnitude of z. The variation in color adds to the three-dimensional visualization of the plots. The color can be changed to be a constant either by using the Plot Editor in the Figure Window (select the edit arrow, click on the figure to open the Property Editor Window, then change the color in the Mesh Properties list), or by using the `colormap(C)` command. In this command `C` is a three-element vector in which the first, second, and third elements specify the intensity of Red, Green, and Blue (RGB) colors, respectively. Each element can be a number between 0 (minimum intensity) and 1 (maximum intensity). Some typical colors are:

  C = [0 0 0] black C = [1 0 0] red C = [0 1 0] green
  C = [0 0 1] blue C = [1 1 0] yellow C = [1 0 1] magenta
  C = [0.5 0.5 0.5] gray

- When the mesh command executes, the grid is on by default. The grid can be turned off with the `grid off` command.
- A box can be drawn around the plot with the `box on` command.
- The `mesh` and `surf` commands can also be used with the form `mesh(Z)` and `surf(Z)`. In this case the values of `Z` are plotted as a function of their addresses in the matrix. The row number is on the x axis and the column number is on the y axis.

There are several additional plotting commands that are similar to the `mesh` and `surf` commands that create plots with different features. Table 10-1 shows a summary of the mesh and surface plotting commands. All the examples in the table are plots of the function $z = 1.8^{-1.5\sqrt{x^2+y^2}}\sin(x)\cos(0.5y)$ over the domain $-3 \le x \le 3$ and $-3 \le y \le 3$.

Table 10-1: Mesh and surface plots

Plot type	Example of plot	Program

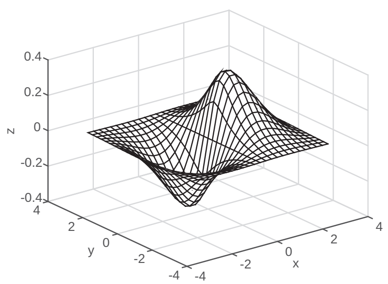
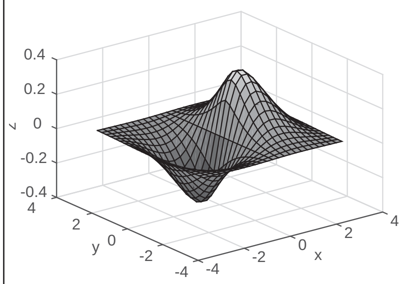
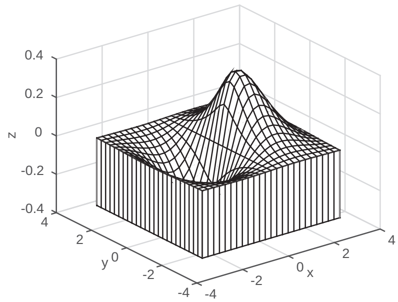
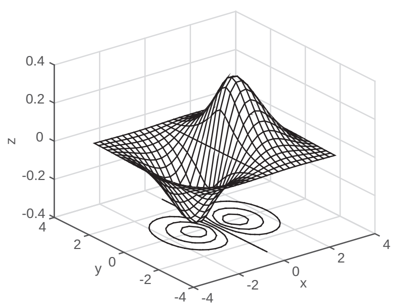
Mesh Plot
 	
Function format:
`mesh(X,Y,Z)`
		`x=-3:0.25:3;`
		`y=-3:0.25:3;`
		`[X,Y] = meshgrid(x,y);`
		`Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(0.5*Y).*sin(X);`
		`mesh(X,Y,Z)`
		`xlabel('x');`
		`ylabel('y')`
		`zlabel('z')`

Table 10-1: Mesh and surface plots (Continued)

Plot type	Example of plot	Program

<u>Surface Plot</u>
 	
Function format:
`surf(X,Y,Z)`
		`x=-3:0.25:3;`
		`y=-3:0.25:3;`
		`[X,Y] = meshgrid(x,y);`
		`Z=1.8.^(-1.5*sqrt(X.^2+`
		`Y.^2)).*cos(0.5*Y).*sin(X);`
		`surf(X,Y,Z)`
		`xlabel('x');`
		`ylabel('y')`
		`zlabel('z')`

<u>Mesh Curtain Plot</u> (draws a curtain around the mesh)
 	
Function format:
`meshz(X,Y,Z)`
		`x=-3:0.25:3;`
		`y=-3:0.25:3;`
		`[X,Y] = meshgrid(x,y);`
		`Z=1.8.^(-1.5*sqrt(X.^2+`
		`Y.^2)).*cos(0.5*Y).*sin(X);`
		`meshz(X,Y,Z)`
		`xlabel('x');`
		`ylabel('y')`
		`zlabel('z')`

<u>Mesh and Contour Plot</u> (draws a contour plot beneath the mesh)
 	
Function format:
`meshc(X,Y,Z)`
		`x=-3:0.25:3;`
		`y=-3:0.25:3;`
		`[X,Y] = meshgrid(x,y);`
		`Z=1.8.^(-1.5*sqrt(X.^2+`
		`Y.^2)).*cos(0.5*Y).*sin(X);`
		`meshc(X,Y,Z)`
		`xlabel('x');`
		`ylabel('y')`
		`zlabel('z')`

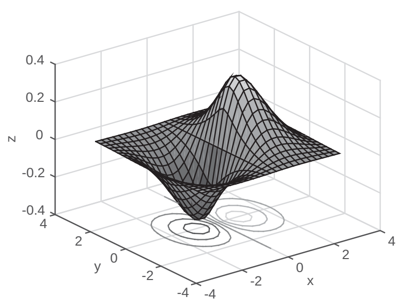
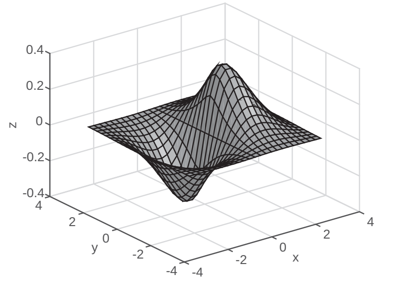
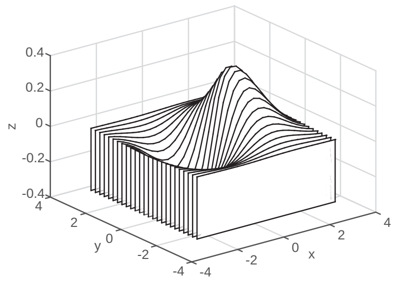
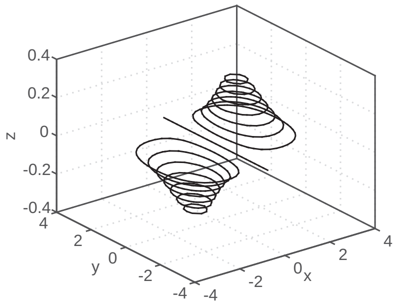
<u>Surface and Contour Plot</u> (draws a contour plot beneath the surface)
 	
Function format:
`surfc(X,Y,Z)`
		`x=-3:0.25:3;`
		`y=-3:0.25:3;`
		`[X,Y] = meshgrid(x,y);`
		`Z=1.8.^(-1.5*sqrt(X.^2+`
		`Y.^2)).*cos(0.5*Y).*sin(X);`
		`surfc(X,Y,Z)`
		`xlabel('x');`
		`ylabel('y')`
		`zlabel('z')`

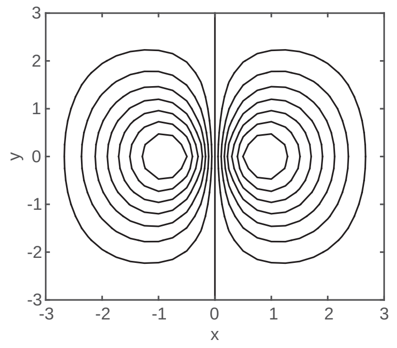
Table 10-1: Mesh and surface plots (Continued)

Plot type	Example of plot	Program

<u>Surface Plot with Lighting</u>
 	
Function format:
`surfl(X,Y,Z)`
		`x=-3:0.25:3;`
		`y=-3:0.25:3;`
		`[X,Y] = meshgrid(x,y);`
		`Z=1.8.^(-1.5*sqrt(X.^2+`
		`Y.^2)).*cos(0.5*Y).*sin(X);`
		`surfl(X,Y,Z)`
		`xlabel('x');`
		`ylabel('y')`
		`zlabel('z')`

<u>Waterfall Plot</u>
(draws a mesh in one direction only)
 	
Function format:
`water-`
`fall(X,Y,Z)`
		`x=-3:0.25:3;`
		`y=-3:0.25:3;`
		`[X,Y] = meshgrid(x,y);`
		`[X,Y] = meshgrid(x,y);`
		`Z=1.8.^(-1.5*sqrt(X.^2+`
		`Y.^2)).*cos(0.5*Y).*sin(X);`
		`waterfall(X,Y,Z)`
		`xlabel('x');`
		`ylabel('y')`
		`zlabel('z')`

<u>3-D Contour Plot</u>
 	
Function format:
`contour3(X,`
`Y,Z,n)`
 	
`n` is the number of contour levels (optional)
		`x=-3:0.25:3;`
		`y=-3:0.25:3;`
		`[X,Y] = meshgrid(x,y);`
		`Z=1.8.^(-1.5*sqrt(X.^2+`
		`Y.^2)).*cos(0.5*Y).*sin(X);`
		`contour3(X,Y,Z,15)`
		`xlabel('x');`
		`ylabel('y')`
		`zlabel('z')`

<u>2-D Contour Plot</u>
(draws projections of contour levels on the *x y* plane)
Function format:
`contour`
`(X,Y,Z,n)`
`n` is the number of contour levels (optional)
		`x=-3:0.25:3;`
		`y=-3:0.25:3;`
		`[X,Y ]=meshgrid(x,y);`
		`[X,Y] = meshgrid(x,y);`
		`Z=1.8.^(-1.5*sqrt(X.^2+`
		`Y.^2)).*cos(0.5*Y).*sin(X);`
		`contour(X,Y,Z,15)`
		`xlabel('x');`
		`ylabel('y')`
		`zlabel('z')`