3D PLOTTING

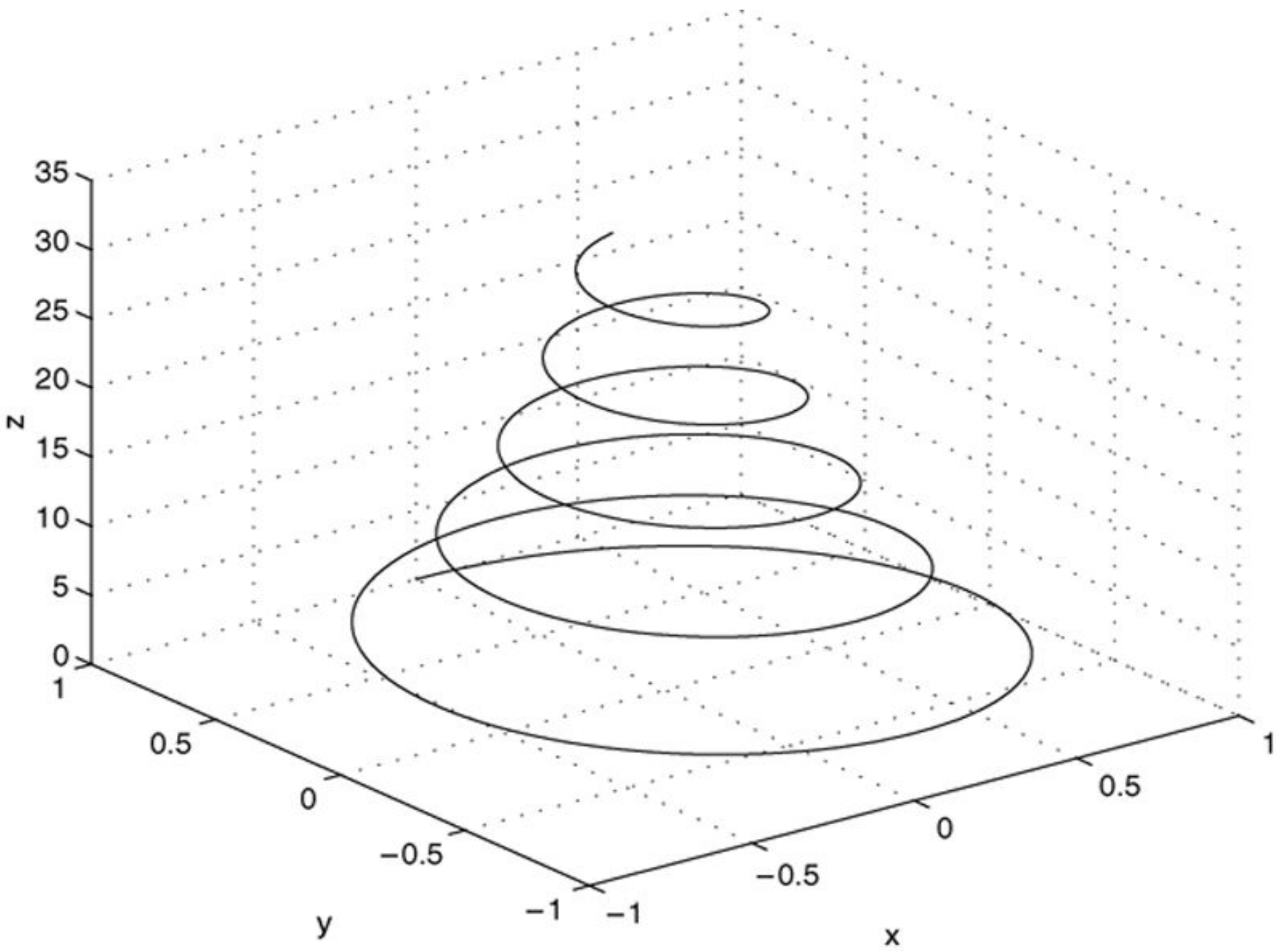
3D line plot

Three-Dimensional Line Plots:

The following program uses the `plot3` function to generate the spiral curve.

```
>>t = [0:pi/50:10*pi];
>>plot3(exp(-0.05*t).*sin(t),
exp(-0.05*t).*cos(t),t))
xlabel('x'),ylabel('y'),zlabel('z')
```

The curve $x = e^{-0.05t} \sin t$, $y = e^{-0.05t} \cos t$, $z = t$ plotted with the `plot3` function.



Three-dimensional plotting functions.

Function	Description
contour(x,y,z)	Creates a contour plot.
mesh(x,y,z)	Creates a 3D mesh surface plot.
meshc(x,y,z)	Same as `mesh` but draws contours under the surface.
meshz(x,y,z)	Same as `mesh` but draws vertical reference lines under the surface.
surf(x,y,z)	Creates a shaded 3D mesh surface plot.
surfc(x,y,z)	Same as `surf` but draws contours under the surface.
[X,Y] = meshgrid(x,y)	Creates the matrices `X` and `Y` from the vectors `x` and `y` to define a rectangular grid.
[X,Y] = meshgrid(x)	Same as `[X,Y]=meshgrid(x,x)`.
waterfall(x,y,z)	Same as `mesh` but draws mesh lines in one direction only.

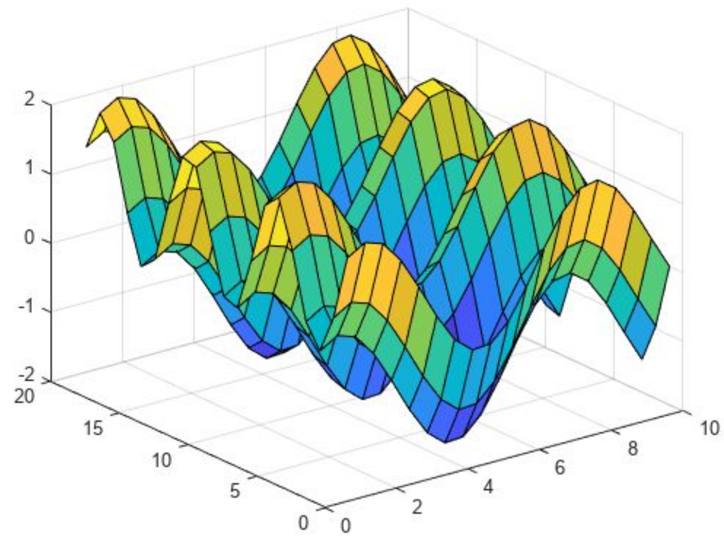
Surf plot

- surf(X,Y,Z) creates a three-dimensional surface plot, which is a three-dimensional surface that has solid edge colors and solid face colors. The function plots the values in matrix Z as heights above a grid in the x-y plane defined by X and Y. The color of the surface varies according to the heights specified by Z.

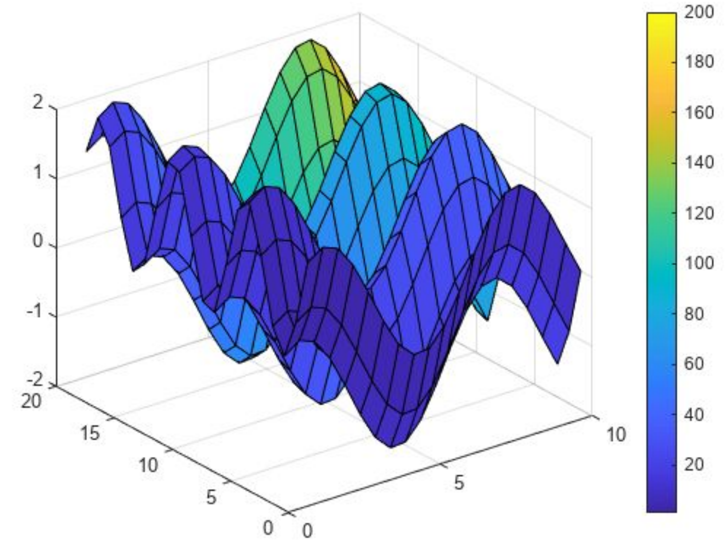
- surf(X,Y,Z,C) additionally specifies the surface color.
- surf(Z) creates a surface plot and uses the column and row indices of the elements in Z as the x- and y-coordinates.

Create three matrices of the same size. Then plot them as a surface. The surface plot uses Z for both height and color.

```
[X,Y] = meshgrid(1:0.5:10,1:20);
Z = sin(X) + cos(Y);
surf(X,Y,Z)
```



```
[X,Y] = meshgrid(1:0.5:10,1:20);
 Z = sin(X) + cos(Y);
C = X.*Y;
surf(X,Y,Z,C)
 colorbar
```



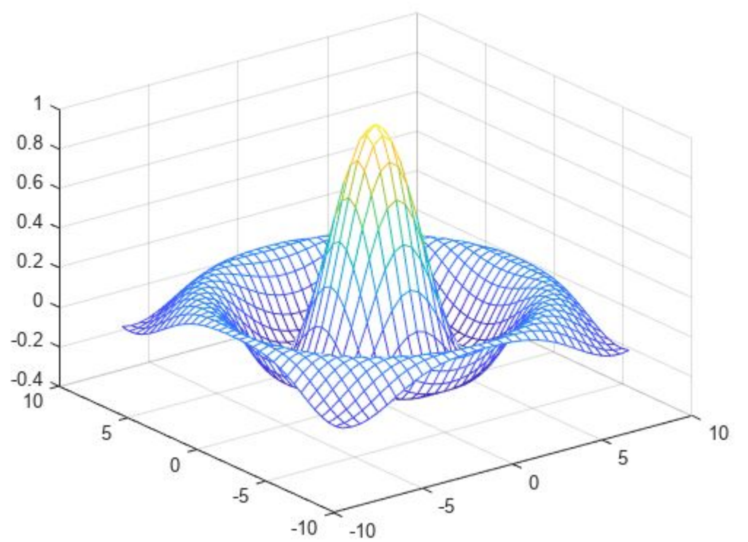
mesh plot

- mesh(X,Y,Z) creates a mesh plot, which is a three-dimensional surface that has solid edge colors and no face colors. The function plots the values in matrix Z as heights above a grid in the x-y plane defined by X and Y. The edge colors vary according to the heights specified by Z.

- mesh(Z) creates a mesh plot and uses the column and row indices of the elements in Z as the x- and y-coordinates.

Create three matrices of the same size. Then plot them as a mesh plot. The plot uses Z for both height and color.

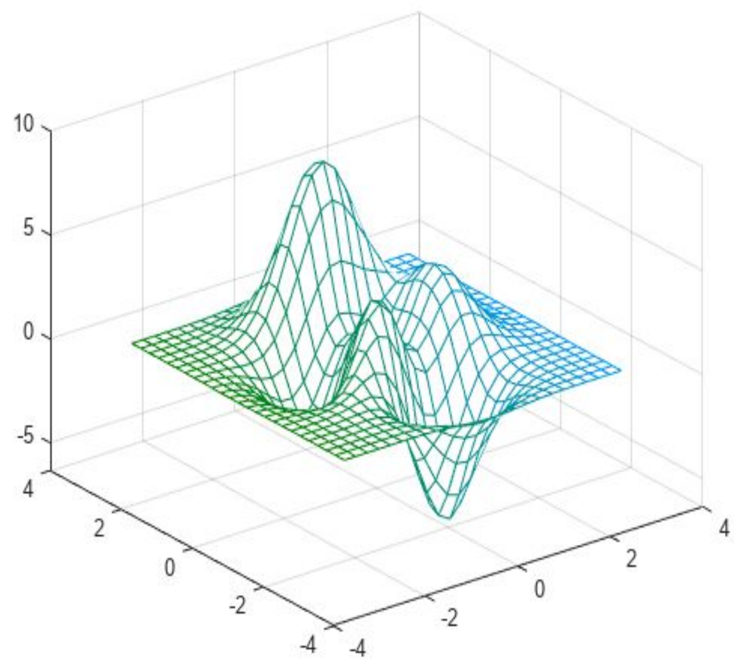
```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
mesh(X,Y,Z)
```



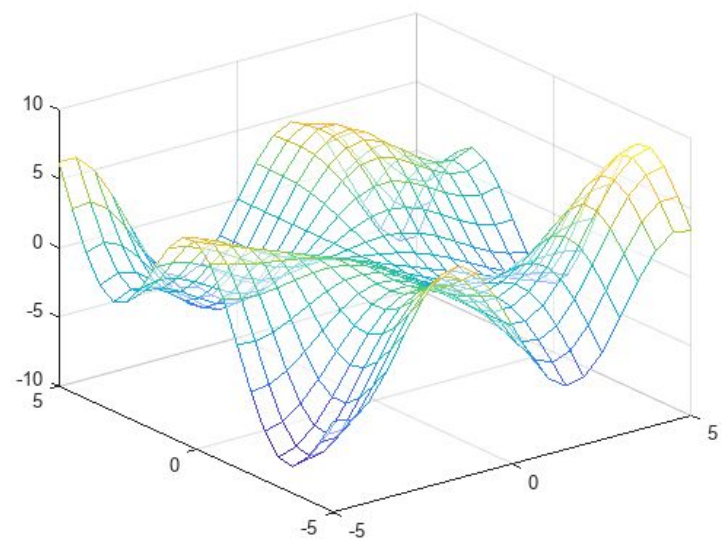
Mesh plot

Specify the colors for a mesh plot by including a fourth matrix input, CO. The mesh plot uses Z for height and CO for color. Specify the colors using *truecolor*, which uses triplets of numbers to stand for all possible colors. When you use truecolor, if Z is m-by-n, then CO is m-by-n-by-3. The first page of the array indicates the red component for each color, the second page indicates the green component, and the third page indicates the blue component.

```
[X,Y,Z] = peaks(25);
CO(:,:,1) = zeros(25);% red
 CO(:,:,2) = ones(25).*linspace(0.5,0.6,25);
% green
 CO(:,:,3) = ones(25).*linspace(0,1,25);
% blue
mesh(X,Y,Z,CO)
```

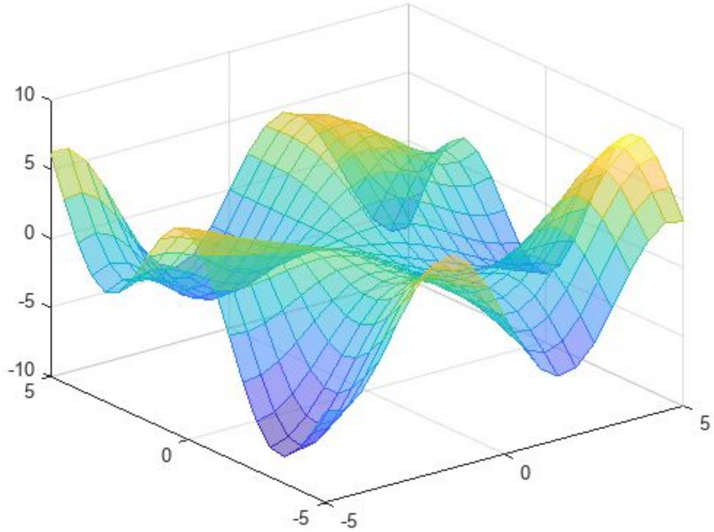


Create a semitransparent mesh surface by specifying the FaceAlpha name-value pair with 0.5 as the value. To allow further modifications, assign the surface object to the variable s.




```
[X,Y] = meshgrid(-5:.5:5);
Z = Y.*sin(X) - X.*cos(Y);
s = mesh(X,Y,Z,'FaceAlpha','0.5')
```

s.FaceColor = 'flat';

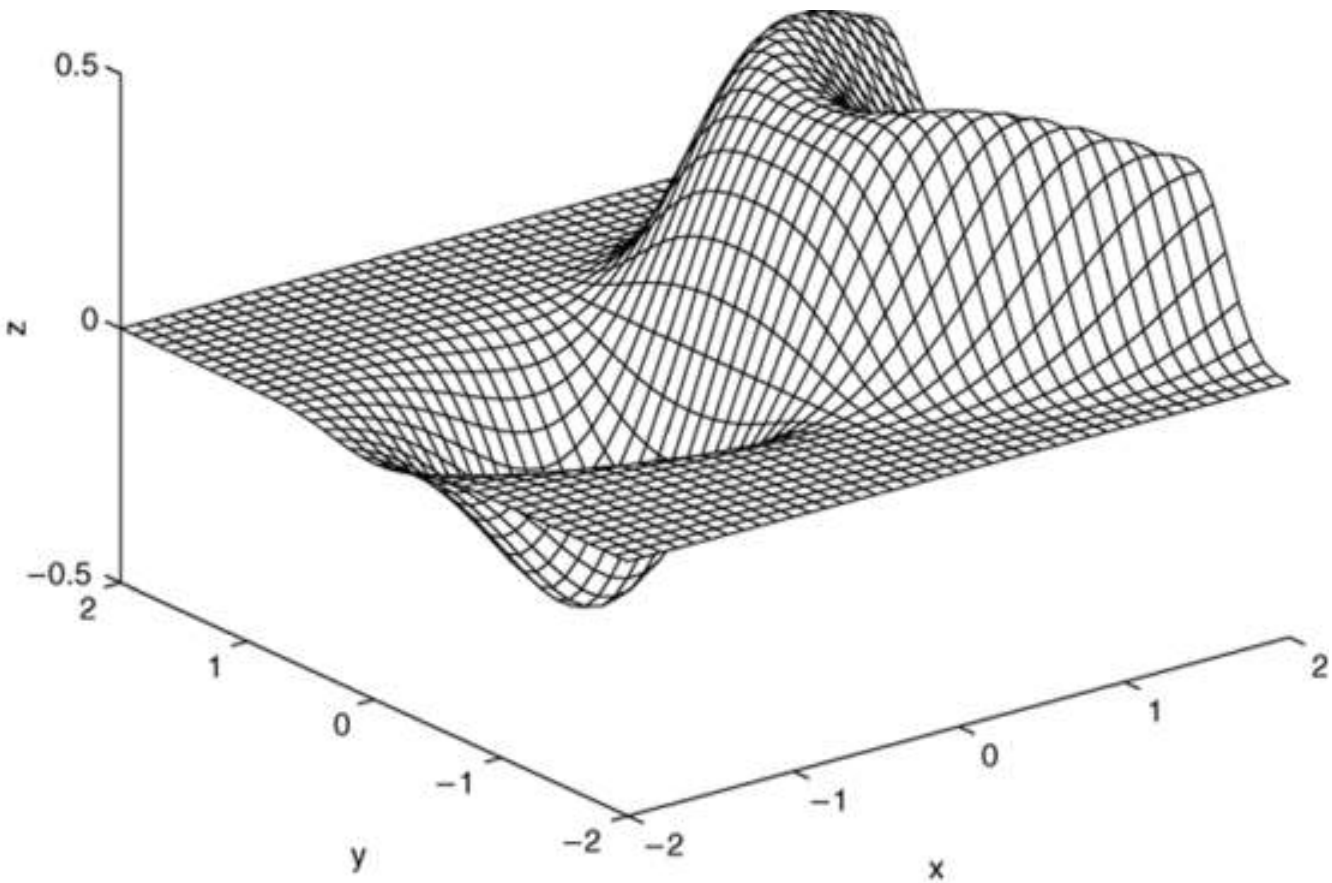


Surface Plots:

The following session shows how to generate the surface plot of the function $z = xe^{-[(x-y2)2+y2]}$, for $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$, with a spacing of 0.1.

```
>>[X,Y] = meshgrid(-2:0.1:2);
>>Z = X.*exp(-((X-Y.^2).^2+Y.^2));
>>mesh(X,Y,Z),xlabel('x'),ylabel('y'),...
zlabel('z')
```

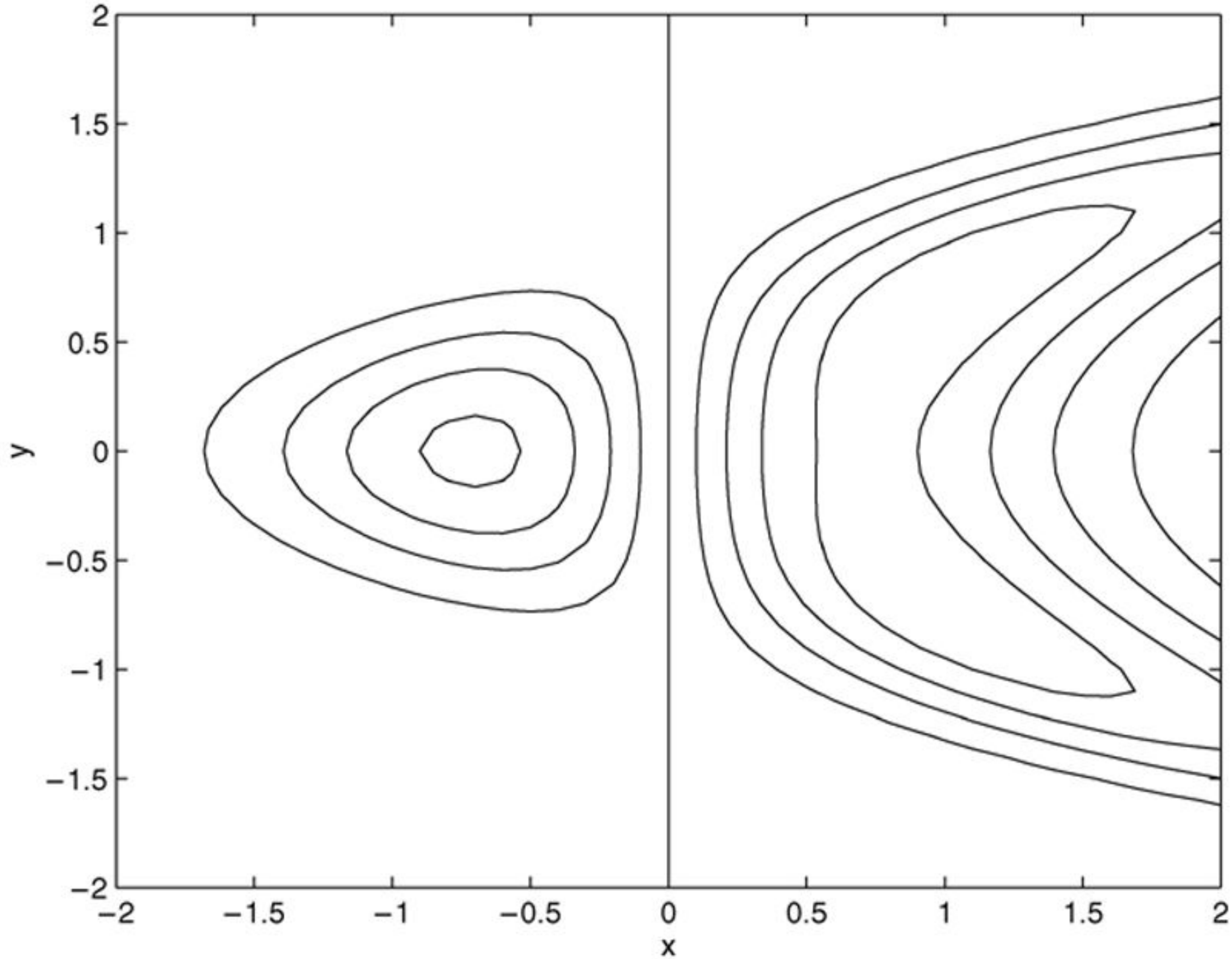
A plot of the surface $z = xe^{-[(x-y2)2+y2]}$ created with the `mesh` function.



The following session generates the contour plot of the function whose surface plot is shown in figure; namely, $z = xe^{-[(x-y^2)^2+y^2]}$, for $-2 \le x \le 2$ and $-2 \le y \le 2$, with a spacing of 0.1.

```
>>[X,Y] = meshgrid(-2:0.1:2);
>>Z = X.*exp(-((X- Y.^2).^2+Y.^2));
>>contour(X,Y,Z),xlabel('x'),ylabel('y')
```

A contour plot of the surface $z = xe^{-[(x-y2)2+y2]}$ created with the `contour` function. F



Contour Plot

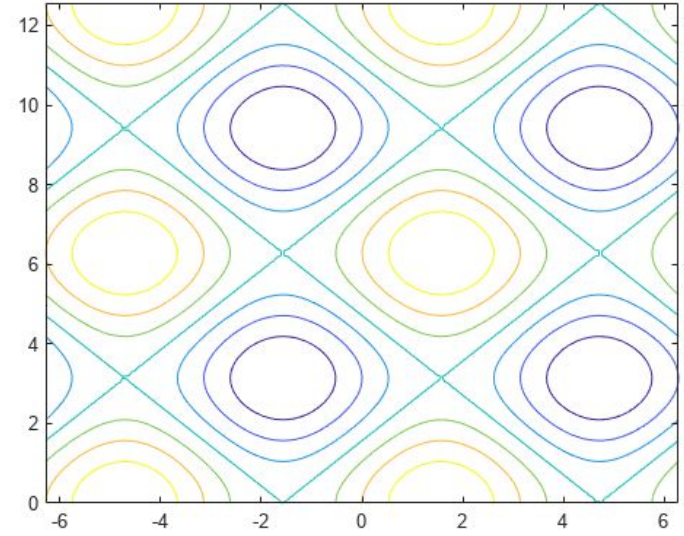
- contour(Z) creates a contour plot containing the isolines of matrix Z, where Z contains height values on the x-y plane.

- contour(X,Y,Z) specifies the x and y coordinates for the values in Z.

- contour(___,levels) specifies the contour lines to display as the last argument in any of the previous syntaxes. Specify levels as a scalar value n to display the contour lines at n automatically chosen levels (heights). To draw the contour lines at specific heights, specify levels as a vector of monotonically increasing values. To draw the contours at one height (k), specify levels as a two-element row vector [k k].

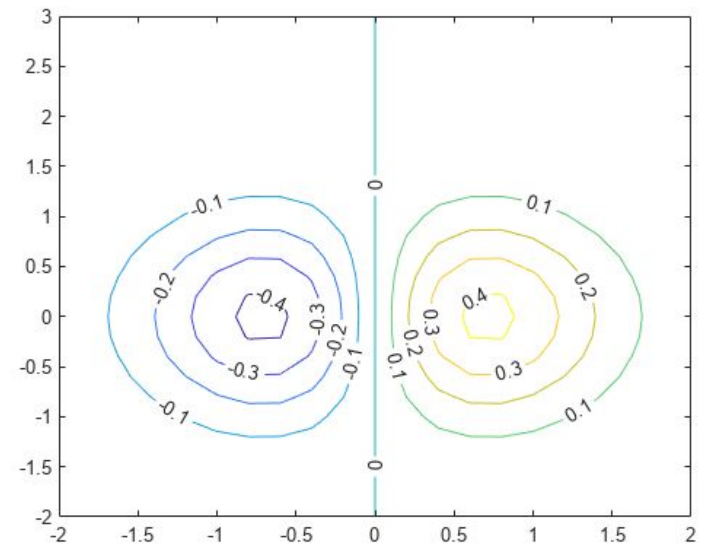
Create matrices X and Y, that define a grid in the x-y plane. Define matrix Z as the heights above that grid. Then plot the contours of Z.

```
x = linspace(-2*pi,2*pi);
y = linspace(0,4*pi);
[X,Y] = meshgrid(x,y);
Z = sin(X)+cos(Y);
contour(X,Y,Z)
```



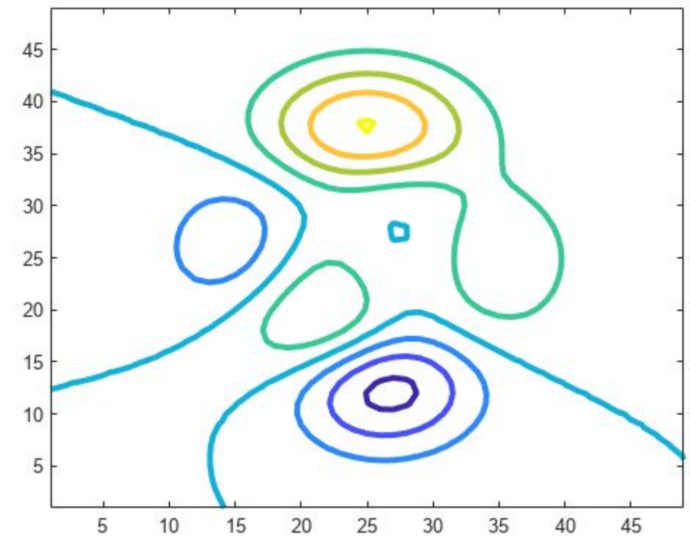
Define Z as a function of two variables, X and Y. Then create a contour plot of that function, and display the labels by setting the ShowText property to 'on'.

```
x = -2:0.2:2;
y = -2:0.2:3;
[X,Y] = meshgrid(x,y);
Z = X.*exp(-X.^2-Y.^2);
contour(X,Y,Z,'ShowText','on')
```



Create a contour plot of the peaks function. Make the contour lines thicker by setting the LineWidth property to 3.

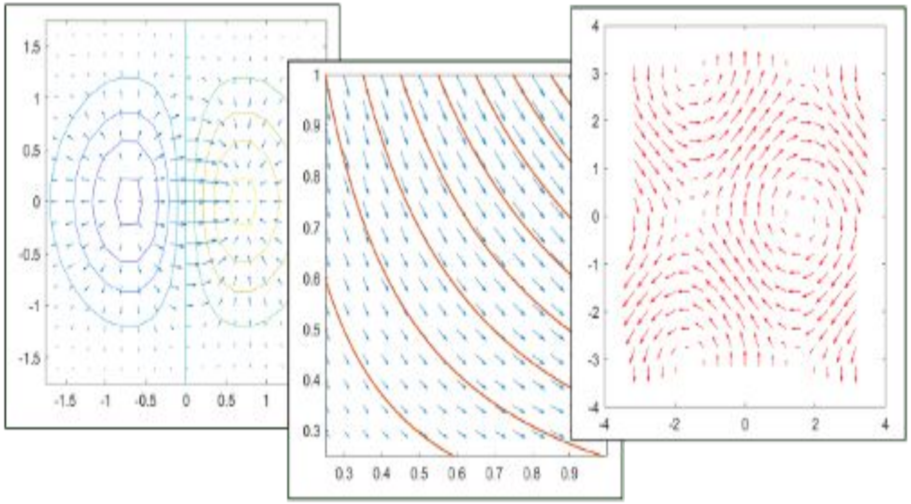
```
Z = peaks;
[M,c] = contour(Z);
c.LineWidth = 3;
```



VECTOR PLOT

Vector fields Plot

- Vector fields can model velocity, magnetic force, fluid motion, and gradients. Visualize vector fields in a 2-D or 3-D view using the quiver, quiver3, and streamline functions. You can also display vectors along a horizontal axis or from the origin.



Quiver plot

- quiver(X,Y,U,V) plots arrows with directional components U and V at the Cartesian coordinates specified by X and Y. For example, the first arrow originates from the point X(1) and Y(1), extends horizontally according to U(1), and extends vertically according to V(1). By default, the quiver function scales the arrow lengths so that they do not overlap.

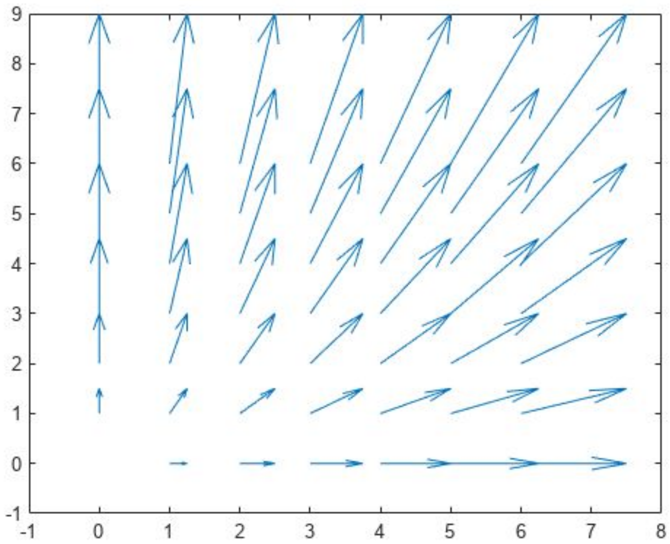
- quiver(X,Y,U,V) plots arrows with directional components U and V at the Cartesian coordinates specified by X and Y. For example, the first arrow originates from the point X(1) and Y(1), extends horizontally according to U(1), and extends vertically according to V(1). By default, the quiver function scales the arrow lengths so that they do not overlap.

- By default, the quiver function shortens arrows so they do not overlap. Disable automatic scaling so that arrow lengths are determined entirely by U and V by setting the scale argument to 0.

- For instance, create a grid of X and Y values using the meshgrid function. Specify the directional components using these values. Then, create a quiver plot with no automatic scaling.

Quiver plot

```
[X,Y] = meshgrid(0:6,0:6);
U = 0.25*X;
V = 0.5*Y;
quiver(X,Y,U,V,0)
```



Plot the gradient and contours of the function.

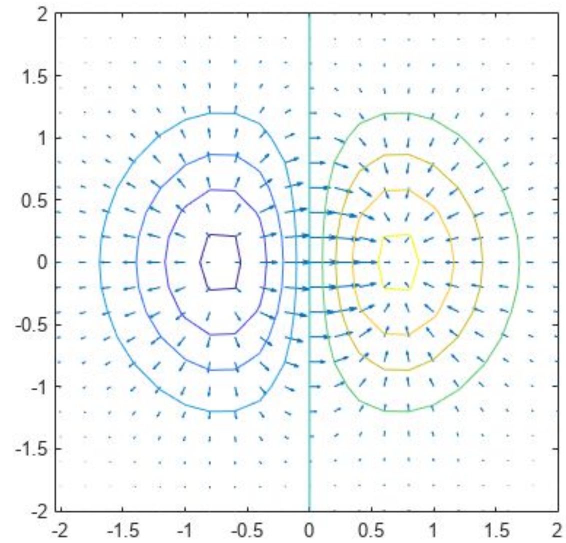
Use the quiver function to plot the gradient and the contour function to plot the contours.

First, create a grid of x- and y-values that are equally spaced. Use them to calculate z. Then, find the gradient of z by specifying the spacing between points.

```
spacing = 0.2;
[X,Y] = meshgrid(-2:spacing:2);
Z = X.*exp(-X.^2 - Y.^2);
[DX,DY] = gradient(Z,spacing);
```

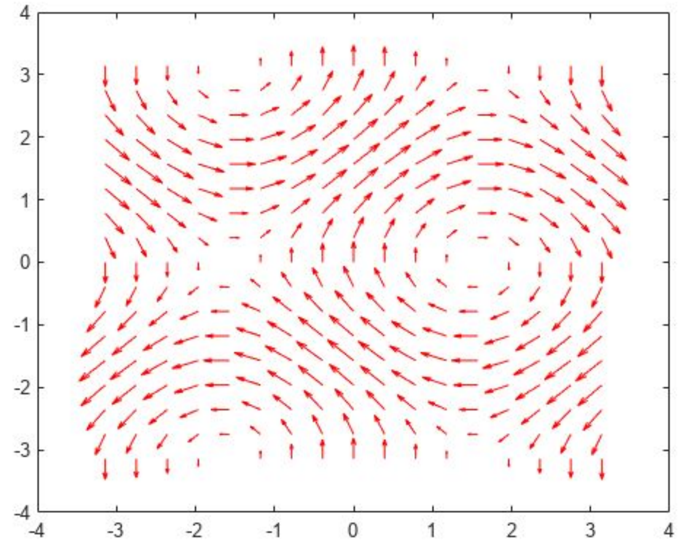
Display the gradient vectors as a quiver plot. Then, display contour lines in the same axes. Adjust the display so that the gradient vectors appear perpendicular to the contour lines by calling axis equal.

```
quiver(X,Y,DX,DY)
hold on
contour(X,Y,Z)
axis equal
hold off
```



Create a quiver plot and specify a color for the arrows.

```
[X,Y] = meshgrid(-pi:pi/8:pi,-pi:pi/8:pi);
U = sin(Y);
V = cos(X);
quiver(X,Y,U,V,'r')
```



Question

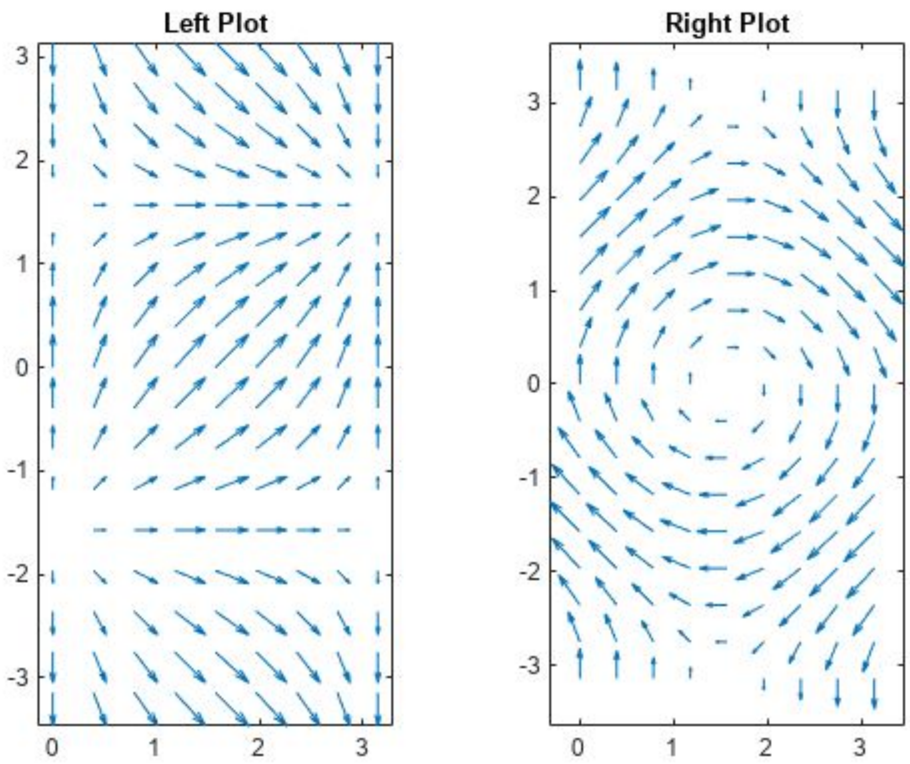
Create a grid of X and values and two sets of U and V directional components.

```
[X,Y] =
   meshgrid(0:pi/8:pi,-pi:pi/
   8:pi);
U1 = sin(X);
V1 = cos(Y);
U2 = sin(Y);
V2 = cos(X);
```

```
tiledlayout(1,2)
ax1 = nexttile;
quiver(ax1,X,Y,U1,V1)
axis equal
title(ax1,'Left Plot')

ax2 = nexttile;
quiver(ax2,X,Y,U2,V2)
axis equal
title(ax2,'Right Plot')
```

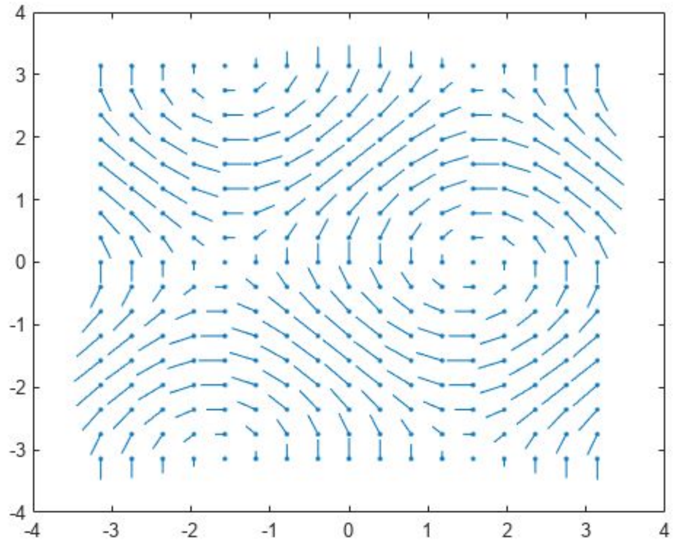
Figure



Create a quiver plot and return the quiver object. Then, remove the arrowheads and add dot markers at the base of each arrow.

```
[X,Y] = meshgrid(-pi:pi/8:pi,-pi:pi/8:pi);
U = sin(Y);
V = cos(X);

q = quiver(X,Y,U,V);
q.ShowArrowHead = 'off';
q.Marker = '.';
```



Feather Plot

feather(U,V) plots arrows originating from the x-axis. Specify the direction of arrows using the Cartesian components U and V, with U indicating the x-components and V indicating the y-components. The nth arrow has its base at n on the x-axis. The number of arrows matches the number of elements in U and V.

feather(Z) plots arrows using the complex values specified by Z, with the real part indicating the x-components and the imaginary part indicating the y-components. This syntax is equivalent to feather(real(Z),imag(Z)).

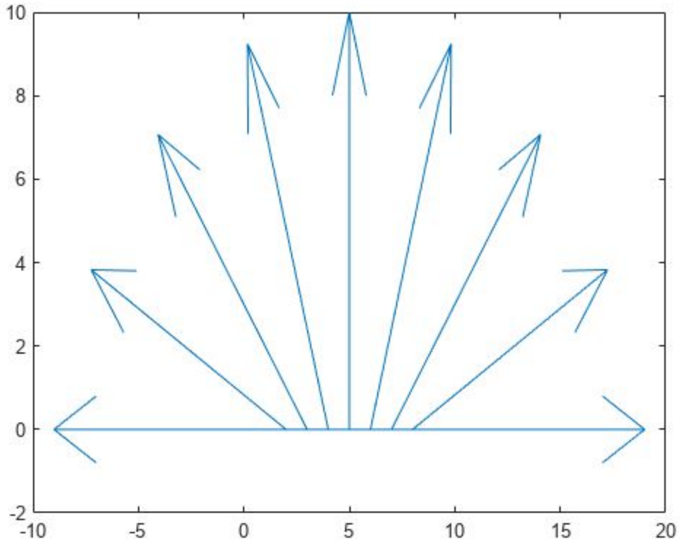
feather(___,LineSpec) sets the line style, marker symbol, and color for the arrows.

feather(ax,___) plots arrows in the specified axes instead of the current axes.

f = feather(___) returns a vector of Line objects with length(U)+1 elements. The first length(U) elements represent individual arrows, and the last element represents a horizontal line along the x-axis. Use these Line objects to control the appearance of the plot after creating it.

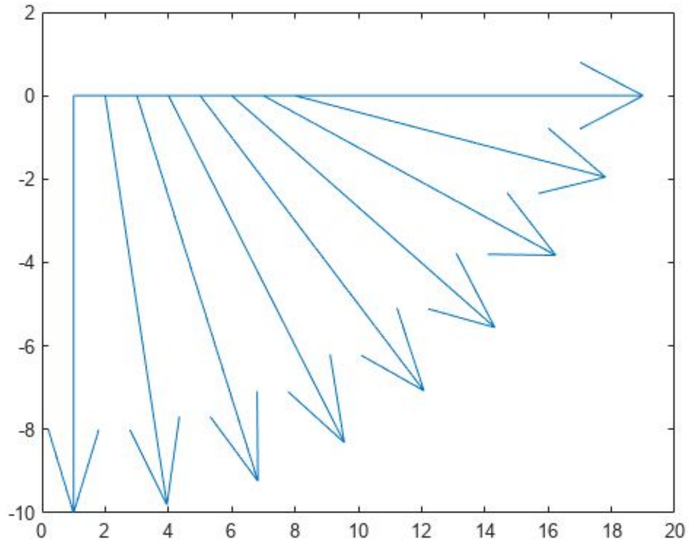
Create a feather plot by specifying the components of each arrow as Cartesian values. The nth arrow originates from n on the x-axis.

```
t = -pi/2:pi/8:pi/2;
u = 10*sin(t);
v = 10*cos(t);
feather(u,v)
```

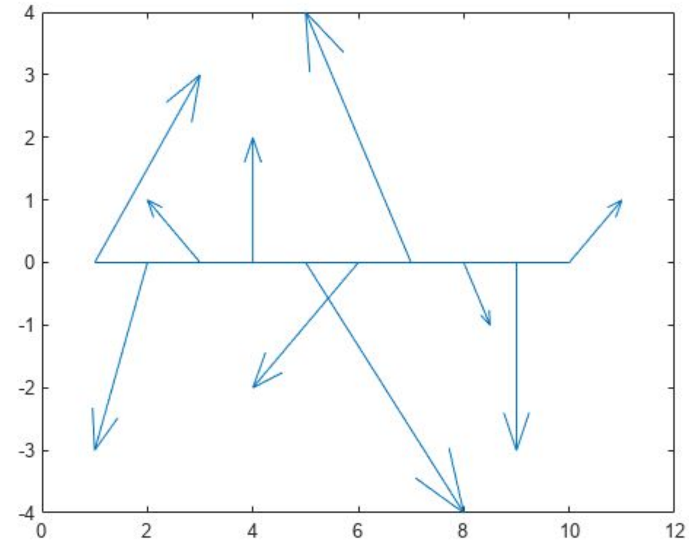


Create a feather plot using polar coordinates by first converting them to Cartesian coordinates. To do this, create vectors with polar coordinates. Convert them to Cartesian coordinates using the pol2cart function. Then, create the plot.

```
th = -pi/2:pi/16:0;
r = 10*ones(size(th));
[u,v] = pol2cart(th,r);
feather(u,v)
```



Create a vector of complex values. Then, display them using a feather plot. The real part determines the *x*-component of each arrow, and the imaginary part determines the *y*-component.



Z = [2+3i -1-3i -1+i 2i 3-4i -2-2i -2+4i 0.5-i -3i 1+i];
feather(Z)