https://www.mathworks.com/help/matlab/ref/meshgrid.html

meshgrid

2-D and 3-D grids

collapse all in page

Syntax

- `[X,Y] = meshgrid(x,y)`
  example
- `[X,Y] = meshgrid(x)`
  example
- `[X,Y,Z] = meshgrid(x,y,z)`
  example
- `[X,Y,Z] = meshgrid(x)`
  example

Description

example

`[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in vectors `x` and `y`. `X` is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`. The grid represented by the coordinates `X` and `Y` has `length(y)` rows and `length(x)` columns.

example

`[X,Y] = meshgrid(x)` is the same as `[X,Y] = meshgrid(x,x)`, returning square grid coordinates with grid size `length(x)`-by-`length(x)`.

example

`[X,Y,Z] = meshgrid(x,y,z)` returns 3-D grid coordinates defined by the vectors `x`, `y`, and `z`. The grid represented by `X`, `Y`, and `Z` has size `length(y)`-by-`length(x)`-by-`length(z)`.

example

`[X,Y,Z] = meshgrid(x)` is the same as `[X,Y,Z] = meshgrid(x,x,x)`, returning 3-D grid coordinates with grid size `length(x)`-by-`length(x)`-by-`length(x)`.

Examples

collapse all

2-D Grid

Open Script

Create 2-D grid coordinates with *x*-coordinates defined by the vector `x` and *y*-coordinates defined by the vector `y`.

```
x = 1:3;
y = 1:5;
[X,Y] = meshgrid(x,y)
X =

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

```
Y =

     1     1     1
     2     2     2
     3     3     3
     4     4     4
     5     5     5
```

Evaluate the expression $x^2 + y^2$ over the 2-D grid.

```
X.^2 + Y.^2
ans =

     2     5    10
     5     8    13
    10    13    18
    17    20    25
    26    29    34
```

Plot Surface

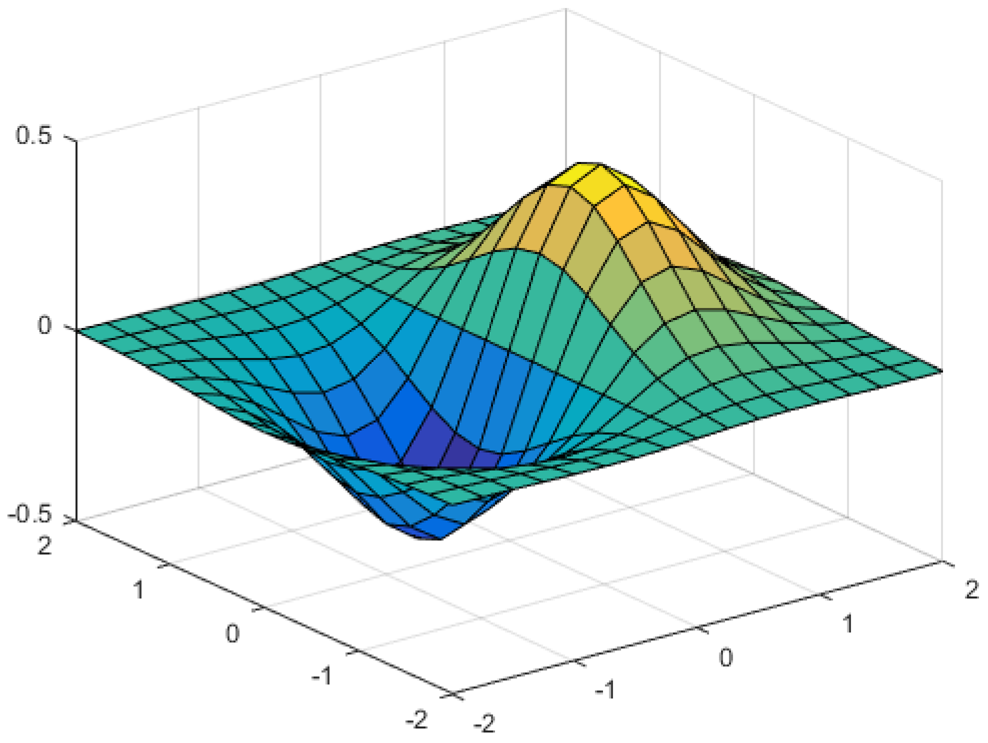
Open Script

Create a 2-D grid with uniformly spaced *x*-coordinates and *y*-coordinates in the interval [-2,2].

```
x = -2:0.25:2;
y = x;
[X,Y] = meshgrid(x);
```

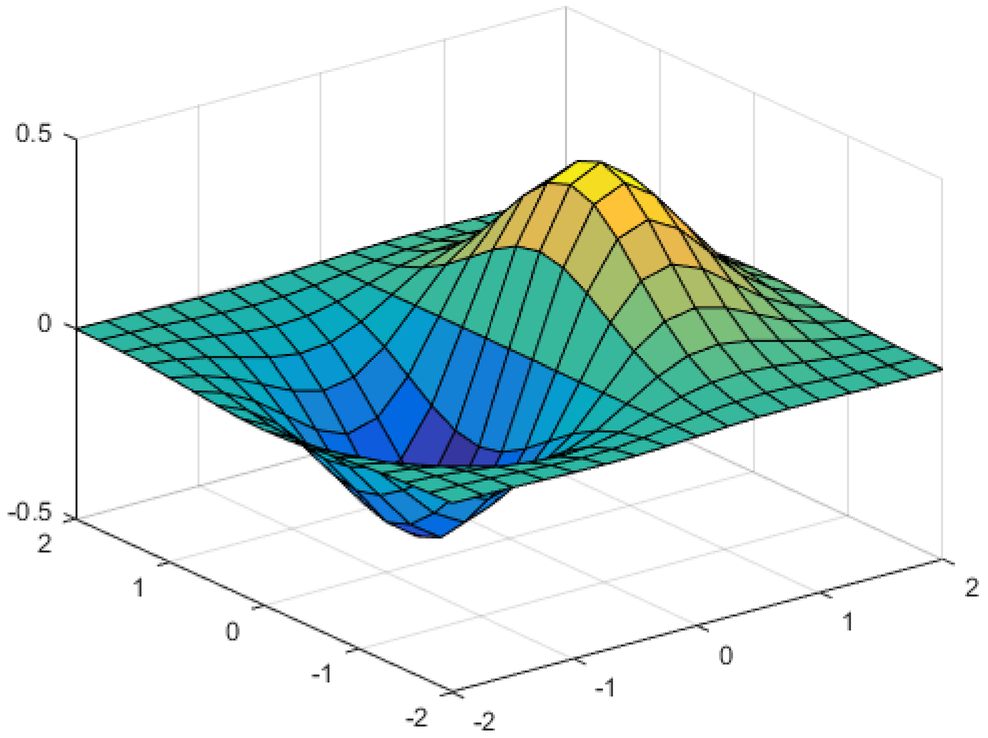
Evaluate and plot the function $f(x,y) = xe^{-x^2-y^2}$ over the 2-D grid.

```
F = X.*exp(-X.^2-Y.^2);
surf(X,Y,F)
```



Starting in R2016b, it is not always necessary to create the grid before operating over it. For example, computing the expression $xe^{-x^2-y^2}$ implicitly expands the vectors `x` and `y`. For more information on implicit expansion, see Array vs. Matrix Operations.

```
surf(x,y,x.*exp(-x.^2-(y').^2))
```



3-D Grid

Open Script

Create 3-D grid coordinates from *x*-, *y*-, and *z*-coordinates defined in the interval [0,6], and evaluate the expression $x^2 + y^2 + z^2$.

```
x = 0:2:6;
y = 0:1:6;
z = 0:3:6;
[X,Y,Z] = meshgrid(x,y,z);
F = X.^2 + Y.^2 + Z.^2;
```

Determine the size of the grid. The three coordinate vectors have different lengths, forming a rectangular box of grid points.

```
gridsize = size(F)
gridsize = 

     7     4     3
```

Use the single-input syntax to generate a uniformly spaced 3-D grid based on the coordinates defined in x. The new grid forms a cube of grid points.

```
[X,Y,Z] = meshgrid(x);
G = X.^2 + Y.^2 + Z.^2;
gridsize = size(G)
gridsize = 

     4     4     4
```

http://www.astro.umd.edu/~cychen/MATLAB/SurfacePlots.html

How to plot Surface Plots

JTK 2010.5.3

Contents

The Function

For our example function, we'll be plotting z = x^2 + y^2. This is essentially a two dimensional parabola.

Generating the X,Y Grid

The first thing to do is to generate an X,Y grid for our plot. Think of this grid as the equivalent of the latitude/longitude on Earth - and that we'll be later be plotting something (say topographic elevation) for each point on this grid.

```
x = -4:1:4;              % The range of x values.
y = -4:1:4;              % The range of y values.
[X,Y] = meshgrid (x,y); % This generates the actual grid of x and y values.
```

Generating the Z Data

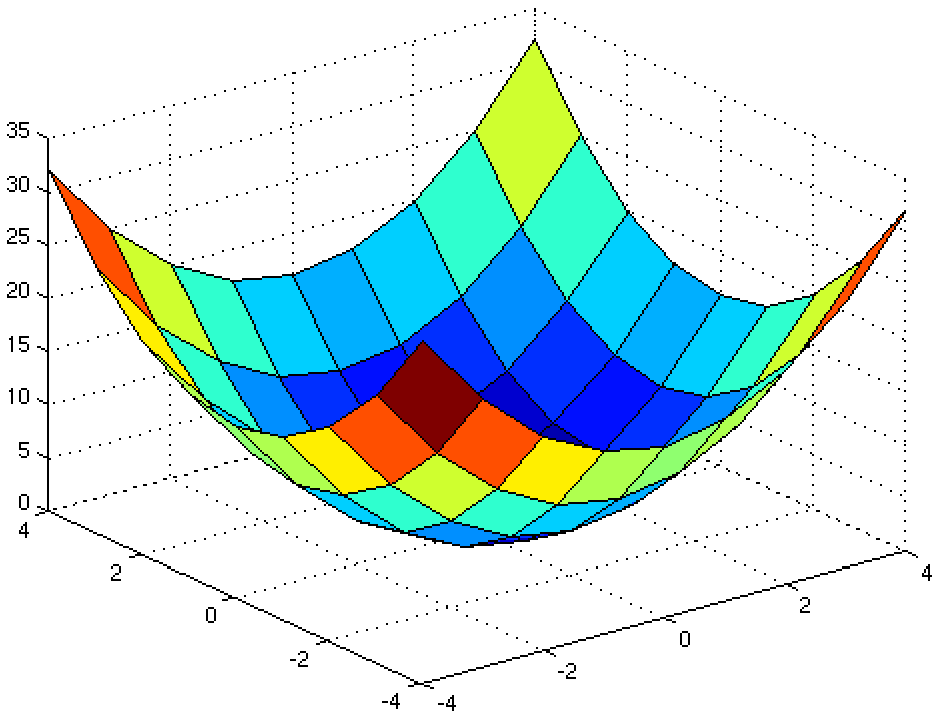
To generate the actual Z data, all we need to do is create a function relating "X" and "Y" (the variables from the meshgrid --- not the ranges, 'x' and 'y').

```
Z=Y.^2+X.^2;                % The function we're plotting.
% Remember to use the correct vector notation for all operations (such as
% using the '.^' operator to do piecewise powers).
```

Generating the Surface Plot

Next, all we need to do is to generate a surface plot!

```
figure(1);                  % Generating a new window to plot in.
surf(X,Y,Z)                 % The surface plotting function.
```

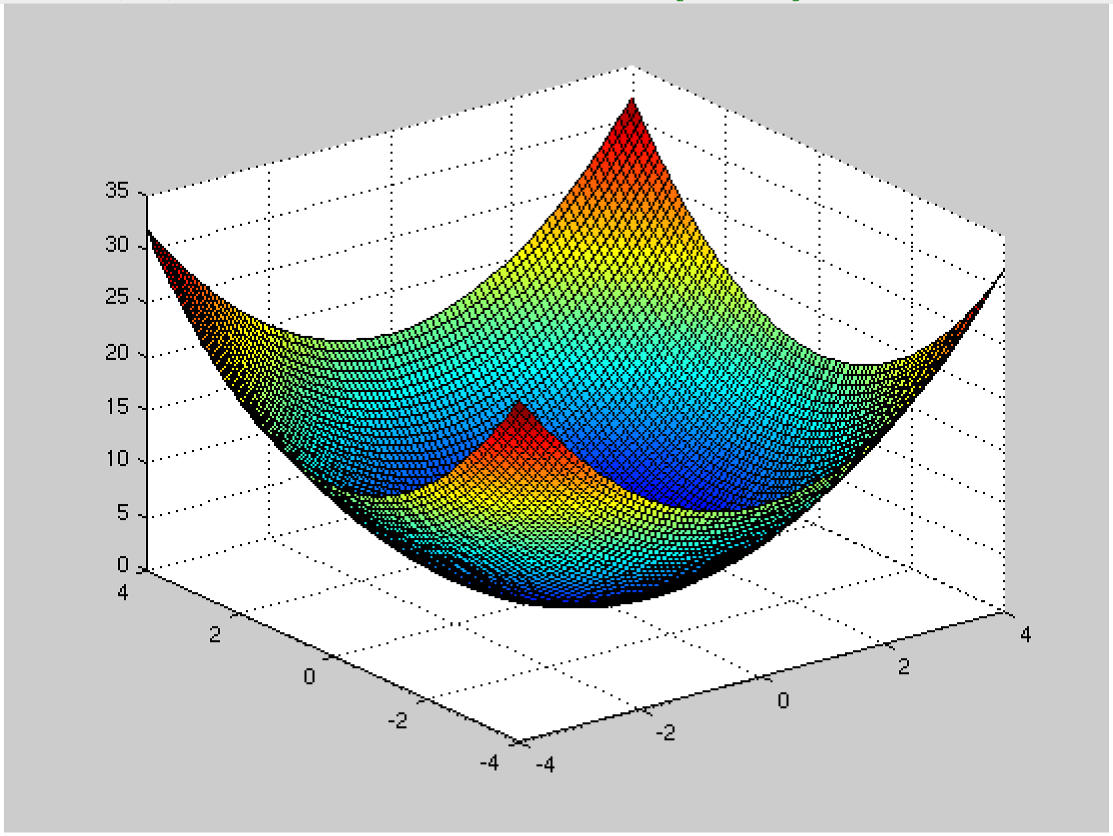


Improving the Resolution of the Surface Plot

To increase the resolution of the surface plot, all we need to do is increase the number of x and y values plotted:

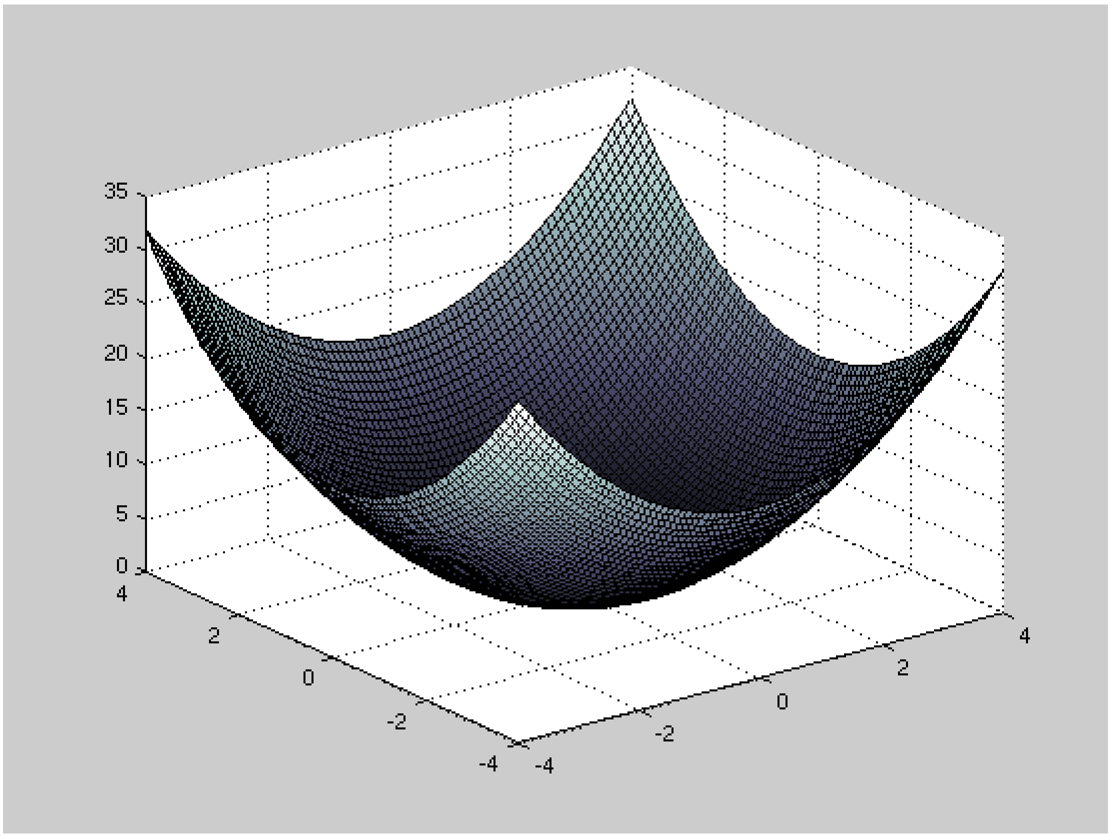
```
figure(2);                    % Generating a new window to plot in.
x1 = -4:.1:4;                 % The range of x values.
y1 = -4:.1:4;                 % The range of y values.
[X1,Y1] = meshgrid (x1,y1); % This generates the actual grid of x and y
values.
Z1=Y1.^2+X1.^2;               % The function we're plotting.
```

```
surf(X1,Y1,Z1)                     % The surface plotting function.
```

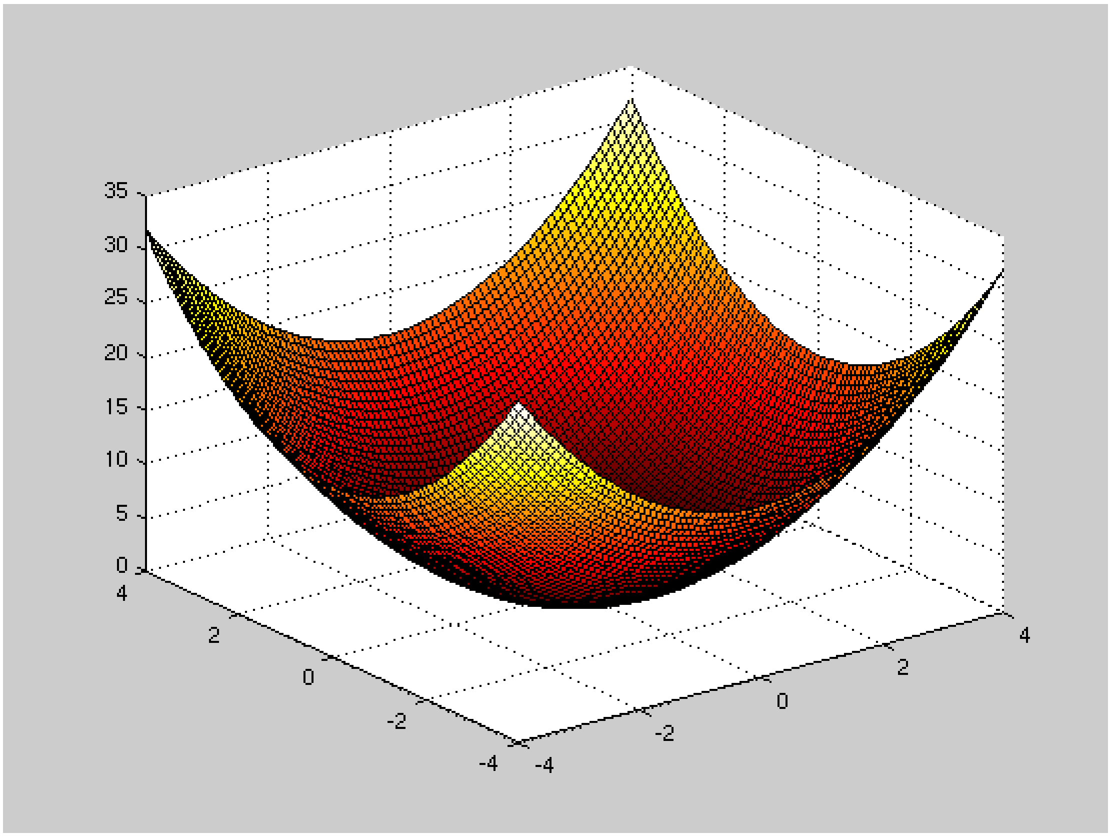


(PRO TIP) Changing the Surface Plot Color

```
%There are a wide array of different colormaps that you can select from,
%just by examining the help file on "colormaps."  Below are two examples.
figure(3); surf(X1,Y1,Z1); colormap(bone)    % Black/white colormap.
figure(4); surf(X1,Y1,Z1); colormap(hot)     % Red/Yellow colormap.
```



35
30
25
20
15
10
5
0
4
2
0
-2
-4
-4
-2
0
2
4

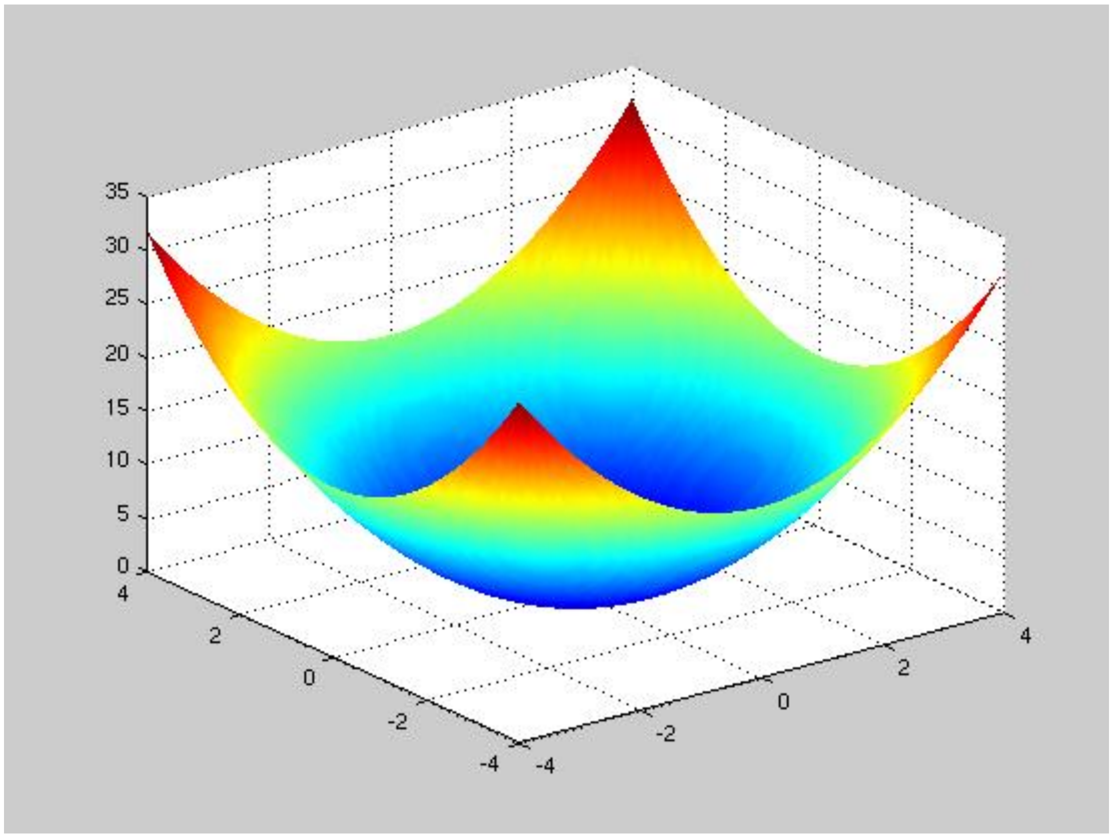


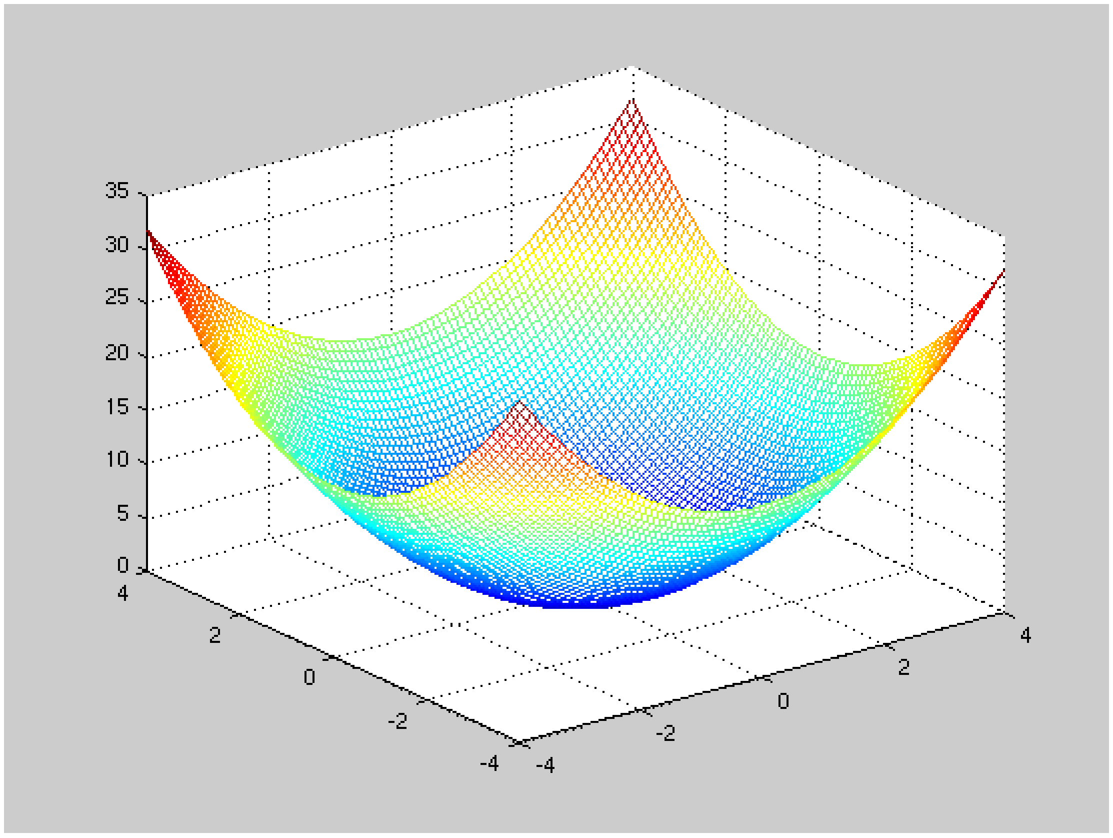
35
30
25
20
15
10
5
0
4
2
0
-2
-4
-4
-2
0
2
4

(PRO TIP) Changing The "Edge" Properties

The "Edges" is the black mesh grid that overlays the 3D surface plot. You can easily change the color (or remove this mesh) by adding a "EdgeColor" qualifier in the "Surf" command prompt.

```
figure(5); surf(X1,Y1,Z1,'EdgeColor','none')
shading interp              % This smooths out the colormap.
% Likewise, you can select to have JUST the edges by using the "mesh"
% plotting function rather than the "surf" function:
figure(6); mesh(X1,Y1,Z1);  % Plotting a mesh plot.
```

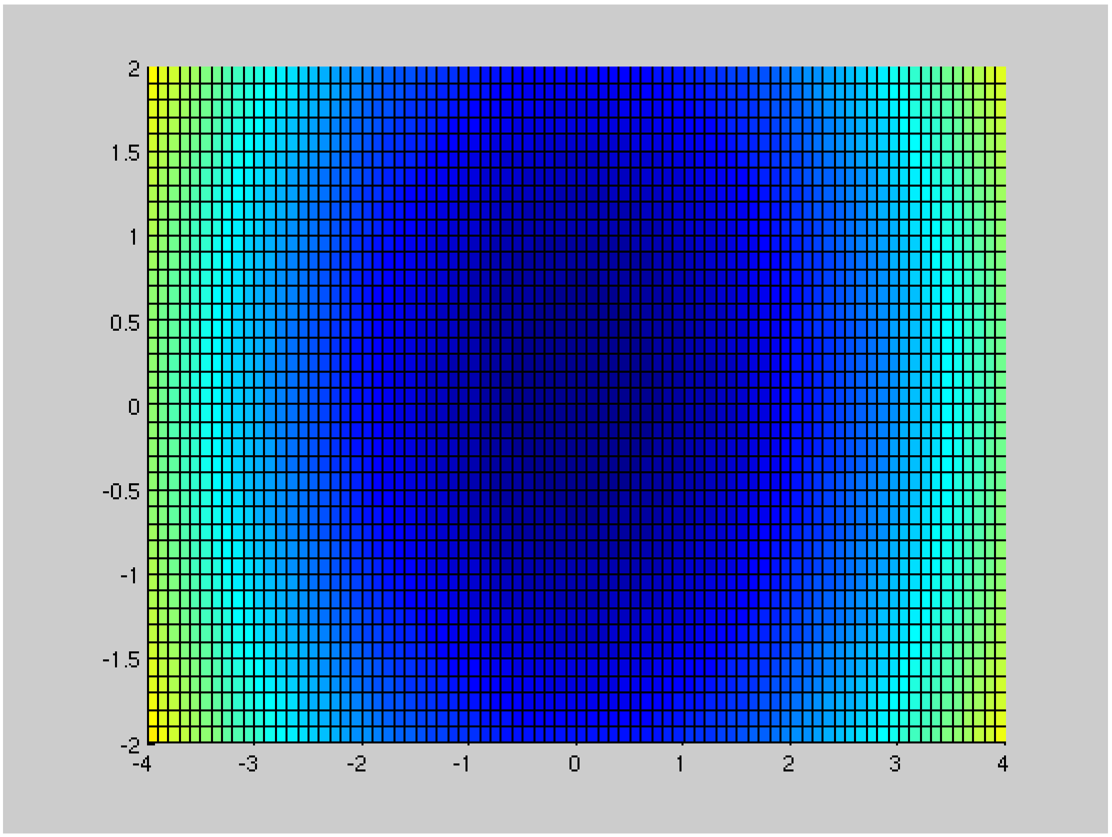

35
30
25
20
15
10
5
0
4
2
0
-2
-4
-4
-2
0
2
4

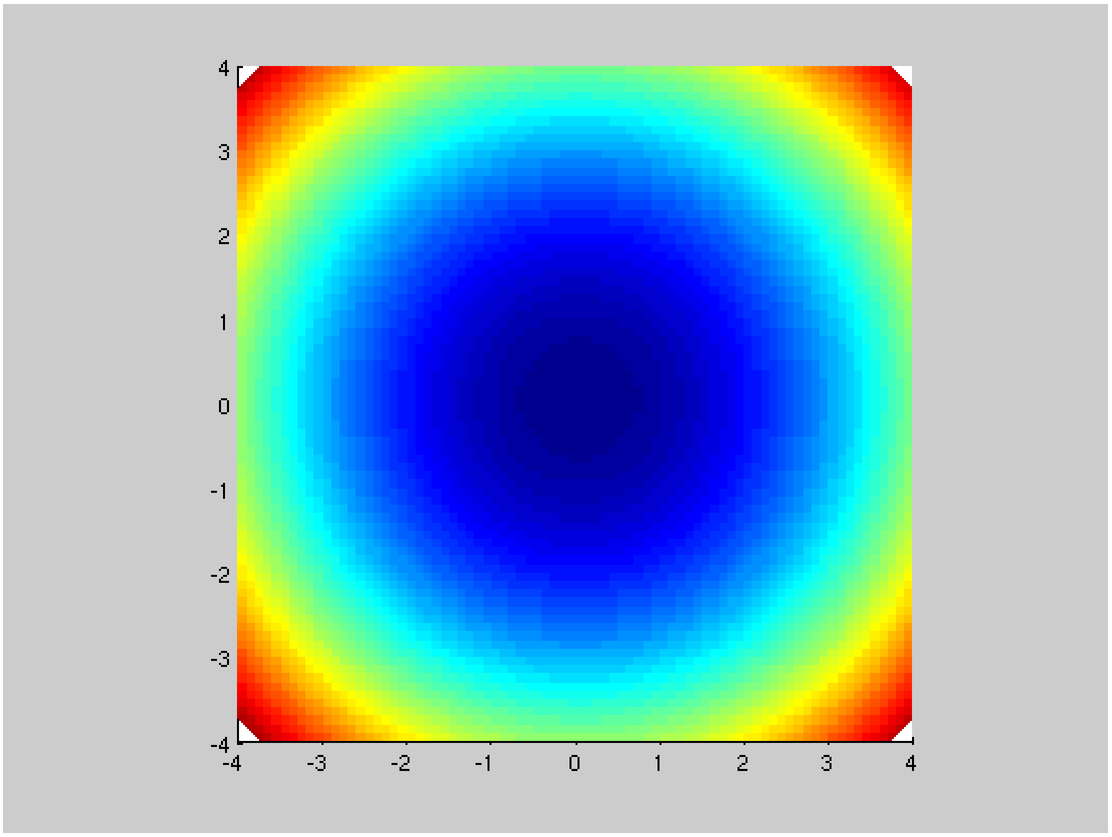

35
30
25
20
15
10
5
0
4
2
0
-2
-4
-4
-2
0
2
4

(PRO TIP) Changing the Plot Axes

To plot custom axes, use the "axis" function:

```
figure(7);
hold on
axis([-4 4 -2 2 -5 30])
surf(X1,Y1,Z1)
hold off
% The axis function is extremely useful, and allows you to do a variety of
% things. To make the scaling of each axis equal, we again use the axis
% function:
figure(8)
hold on
axis([-4 4 -2 2 -5 30])
axis equal
surf(X1,Y1,Z1,'EdgeColor','none')
hold off
% To rotate the figure, you can simply use the "rotate3d" function to make
% the plots interactive.  Once activated, you can then click and drag to
% rotate the plot.
rotate3d on
```


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-4
-3
-2
-1
0
1
2
3
4


4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4