MATLAB Lecture 9 –Graphics -Surface

绘图

Ref: MATLAB→3-D Visualization→Creating 3-D Graphs

- **Vocabulary:**

plot 绘图	surface 曲面
mesh 网格	grid 格子
illustrate 图解	region 区域
polygon 多边形	peak 顶点
tick 记号	helix 螺旋
magenta 洋红色	cyan 青色
gray 灰色	aquamarine 碧绿色

- **Some functions**

Plot3 mesh surf meshgrid fill3 fill *axis square *cylinder *sphere *comet3

- **3-D Graphs**

  ✧ **A Typical 3-D Graph**

  This table illustrates typical steps involved in producing 3-D scenes containing either data graphs or models of 3-D objects.

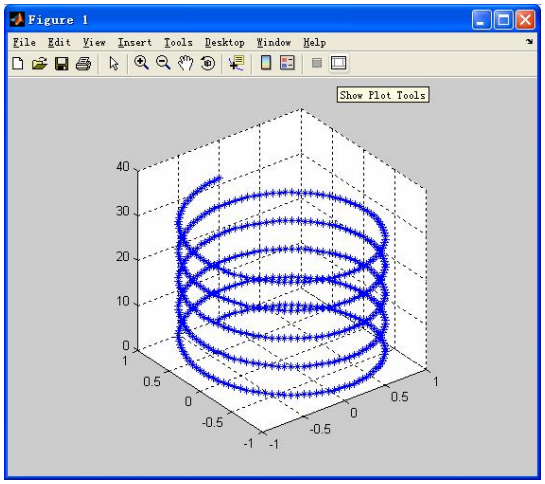
Step	Typical Code
1. Prepare your data.	Z = peaks(20);
2. Select window and position plot region within window.	figure(1) subplot(2,1,2)
3. Call 3-D graphing function.	h = surf(Z);
4. Set colormap and shading algorithm.	colormap hot shading interp set(h,'EdgeColor','k')
5. Add lighting.	light('Position',[-2,2,20]) lighting phong material([0.4,0.6,0.5,30]) set(h,'FaceColor',[0.7 0.7 0],... 'BackFaceLighting','lit')
6. Set viewpoint.	view([30,25]) set(gca,'CameraViewAngleMode','Manual')
7. Set axis limits and tick marks.	axis([5 15 5 15 -8 8]) set(gca,'ZTickLabel','Negative||Positive')
8. Set aspect ratio.	set(gca,'PlotBoxAspectRatio',[2.5 2.5 1])
9. Annotate the graph with axis labels, legend, and text.	xlabel('X Axis') ylabel('Y Axis') zlabel('Function Value') title('Peaks')

10. Print graph.	set(gcf,'PaperPositionMode','auto') print -dps2

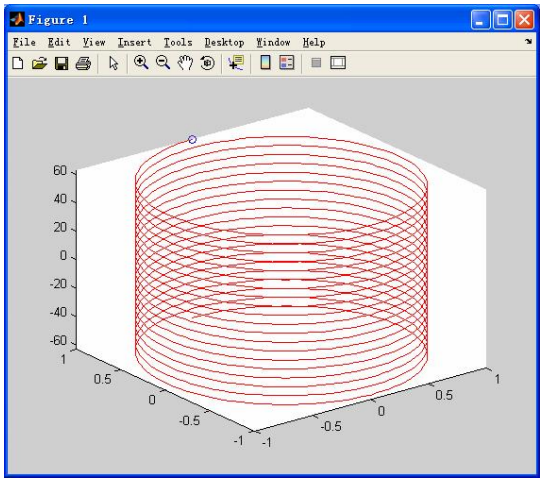
✧ Line Plots of 3-D Data

The 3-D analog of the plot function is plot3. If x, y, and z are three vectors of the same length, plot3(x,y,z) generates a line in 3-D through the points whose coordinates are the elements of x, y, and z and then produces a 2-D projection of that line on the screen.

```
>> t = 0:pi/50:10*pi;
>> plot3(sin(t),cos(t),t, '*-b')
>> axis square; grid on
```



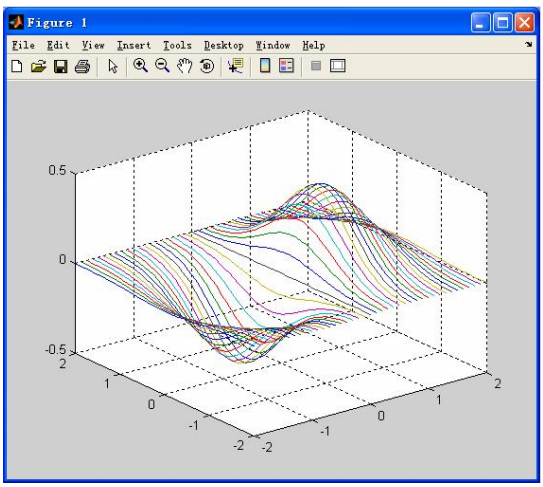

```
>> t=-20*pi:pi/50:20*pi;
>> comet3(sin(t), cos(t),t)
```



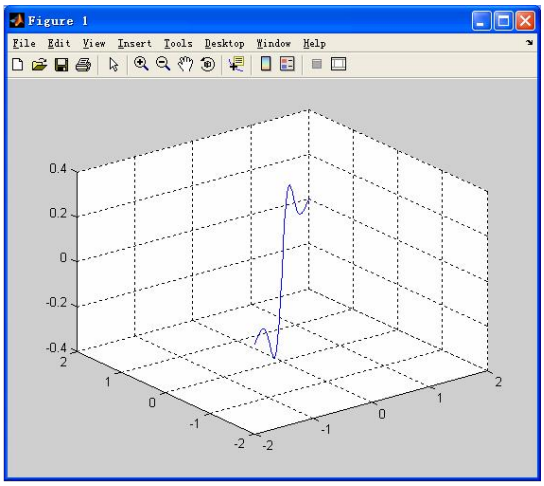
✓ Plotting Matrix Data

If the arguments to plot3 are matrices of the same size, MATLAB plots lines obtained from the columns of X, Y, and Z.

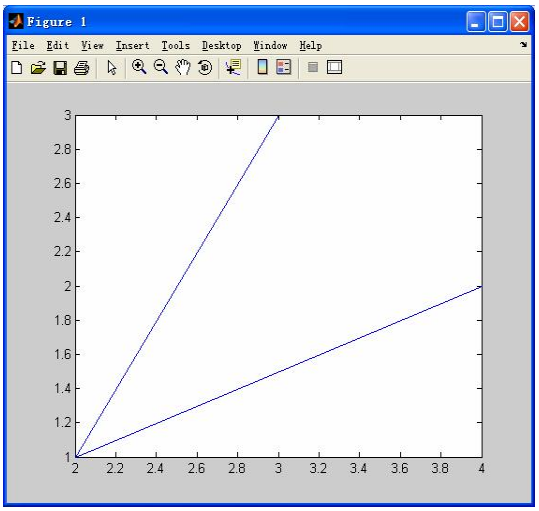
```
>> [X,Y] = meshgrid([-2:0.1:2]);
>> Z = X.*exp(-X.^2-Y.^2);
>> plot3(X,Y,Z)
>> grid on
```



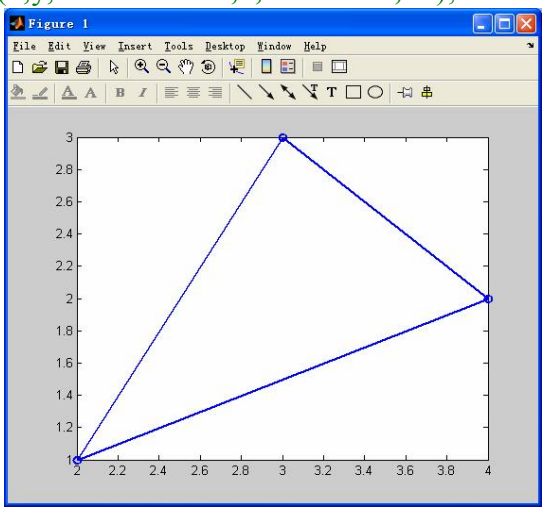

```
>> X=[-2:0.1:2]; Y=[-2:0.1:2];
>> Z = X.*exp(-X.^2-Y.^2);
>> plot3(X,Y,Z)
>> grid on
```



```
>> x=[4 2 3]; y=[2 1 3];
>> plot(x,y)
```




```
>> x=[4 2 3 4]; y=[2 1 3 2];
>> plot(x,y, 'LineWidth',2, 'Marker','o');
```



✧ Fills polygon

fill3(X,Y,Z,C) fills the 3-D polygon defined by vectors X, Y and Z with the color specified by C. The vertices of the polygon are specified by triples of components of X, Y and Z. If necessary, the polygon is closed by connecting the last vertex to the first.

If C is a single character string chosen from the list 'r','g','b', 'c','m','y','w','k', or an RGB row vector triple, [r g b], the polygon is filled with the constant specified color.

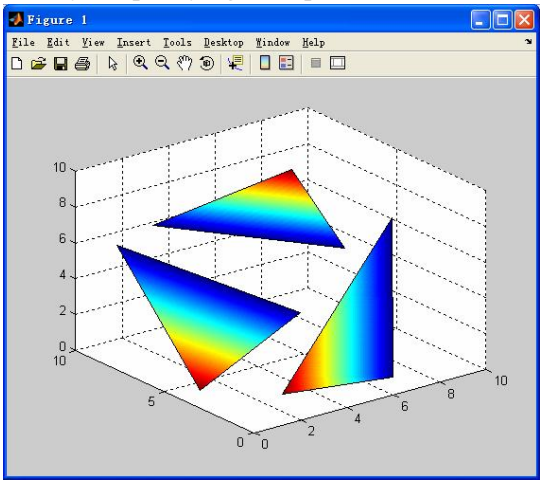
If C is a vector the same length as X, Y and Z, its elements are scaled by CAXIS and used as indices into the current COLORMAP to specify colors at the vertices; the color within the polygon is obtained by bilinear interpolation in the vertex colors.

If X, Y and Z are matrices the same size, one polygon per column is drawn. In this case, C is a row vector for "flat" polygon colors, and C is a matrix for "interpolated" polygon colors.

If any of X, Y or Z is a matrix, and the others are column vectors with the same number of rows, the column vector arguments are replicated to produce matrices of the required size.

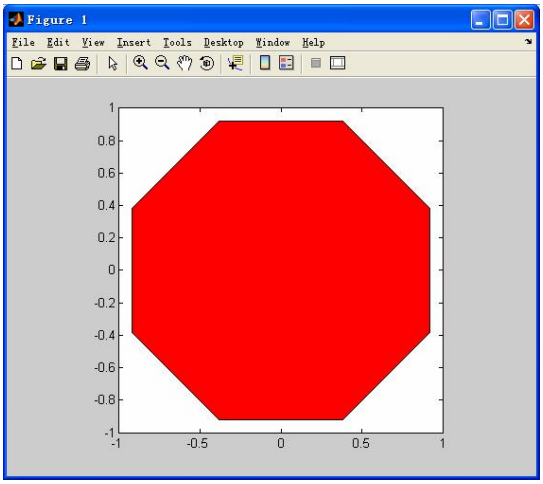
fill3(X1,Y1,Z1,C1,X2,Y2,Z2,C2,...) is another way of specifying multiple filled areas.

```
>> x=[2 1 2; 9 7 1; 6 7 0];
>> y=[1 7 0; 4 7 9; 0 4 3];
>> z=[1 8 6; 7 9 6; 1 6 1];
>> c=[1 0 0; 0 1 0; 0 0 1];
>> fill3(x, y, z, c);
>> grid on
```



fill(X,Y,C) fills the 2-D polygon defined by vectors X and Y with the color specified by C. The vertices of the polygon are specified by pairs of components of X and Y. If necessary, the polygon is closed by connecting the last vertex to the first.

```
>> t = (1/16:1/8:1)'*2*pi;
>> x = sin(t); y = cos(t);
>> fill(x,y,'r')
>> axis square
```



✧ Mesh and Surface Plots

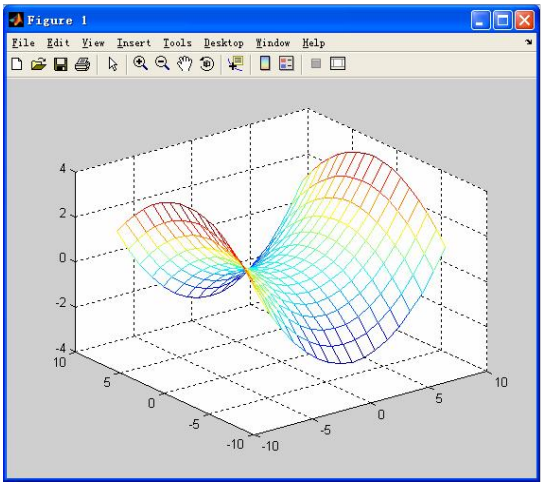
✓ Mesh Plots

mesh(X,Y,Z) draws a wireframe mesh with color determined by Z so color is proportional to surface height. If X and Y are vectors, length(X) = n and length(Y) = m, where [m,n] = size(Z). In this case, $(X(j),Y(i),Z(i,j))$ are the intersections of the wireframe grid lines; X and Y correspond to the columns and rows of Z, respectively. If X and Y are matrices, $(X(i,j),Y(i,j),Z(i,j))$ are the intersections of the wireframe grid lines.

mesh(Z) draws a wireframe mesh using X = 1:n and Y = 1:m, where [m,n] = size(Z). The height, Z, is a single-valued function defined over a rectangular grid. Color is proportional to surface height.

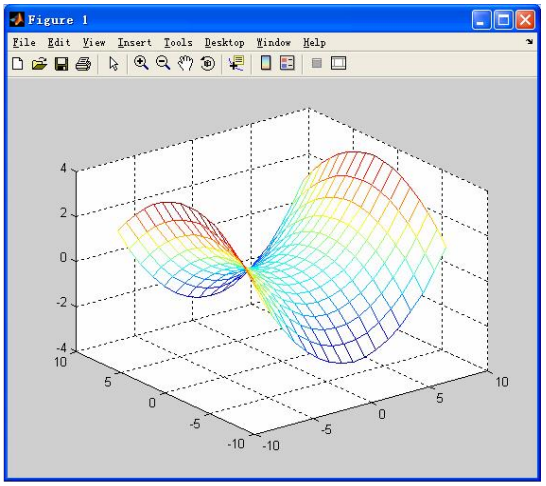
mesh(...,C) draws a wireframe mesh with color determined by matrix C. MATLAB performs a linear transformation on the data in C to obtain colors from the current colormap. If X, Y, and Z are matrices, they must be the same size as C.

```
>> x=-8:8; y=-8:8; [X, Y]=meshgrid(x,y);
>> Z=[X.^2/4^2-Y.^2/5^2];
>> mesh(X,Y,Z)
```

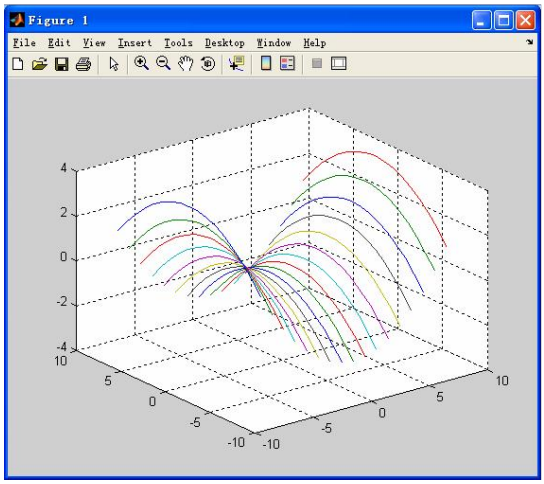



```
>> x=-8:8; y=-8:8; [X, Y]=meshgrid(x,y);
>> Z=[X.^2/4^2-Y.^2/5^2];
```

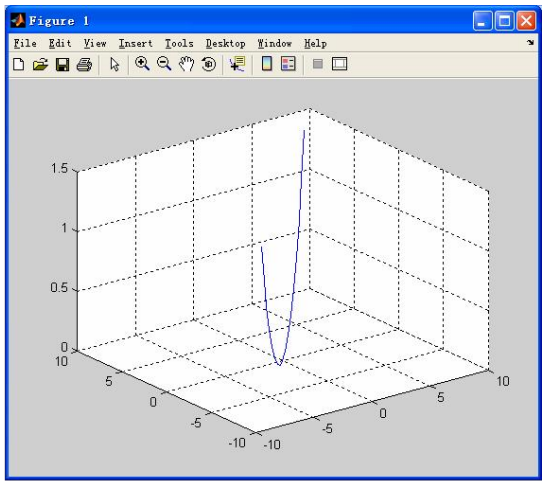
```
>> mesh(x,y, Z)
```



```
>> x=-8:8; y=-8:8; [X, Y]=meshgrid(x,y);
>> Z=[X.^2/4^2-Y.^2/5^2];
>> plot3(X,Y,Z)
>> grid on
```

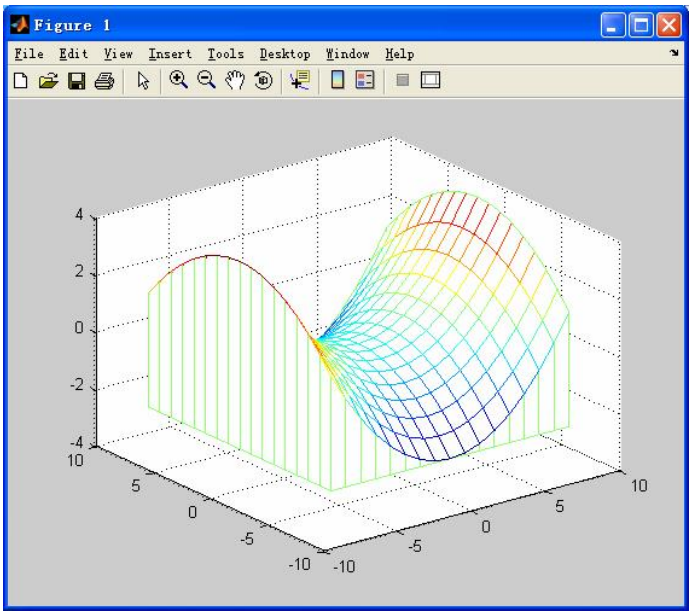



```
>> x=-8:8; y=-8:8;
>> z=[x.^2/4^2-y.^2/5^2];
>> plot3(x,y, z)
>> grid on
```



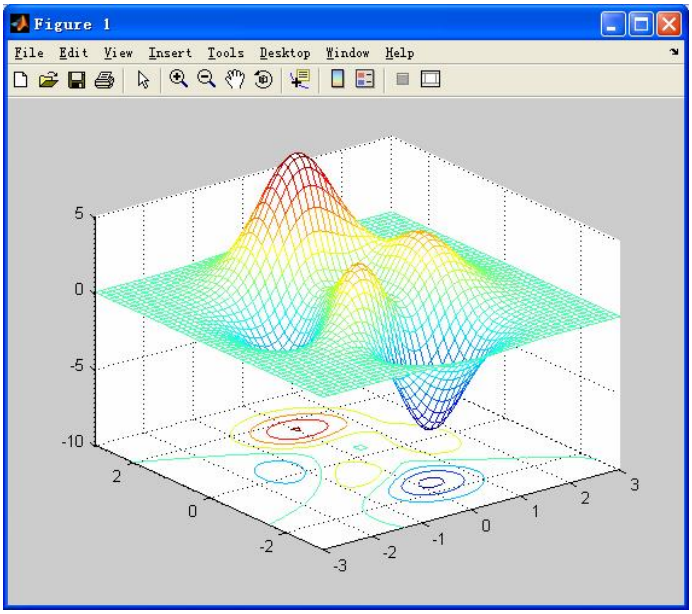
meshz(...) is the same as mesh(...) except that a "curtain" or reference plane is drawn beneath. This routine only works for surfaces defined on a rectangular grid. The matrices X and Y define the axis limits only.

```
>> meshz(X,Y,Z)
```



meshc(...) is the same as mesh(...) except that a contour plot is drawn beneath the mesh. Because CONTOUR does not handle irregularly spaced data, this routine only works for surfaces defined on a rectangular grid. The matrices or vectors X and Y define the axis limits only.

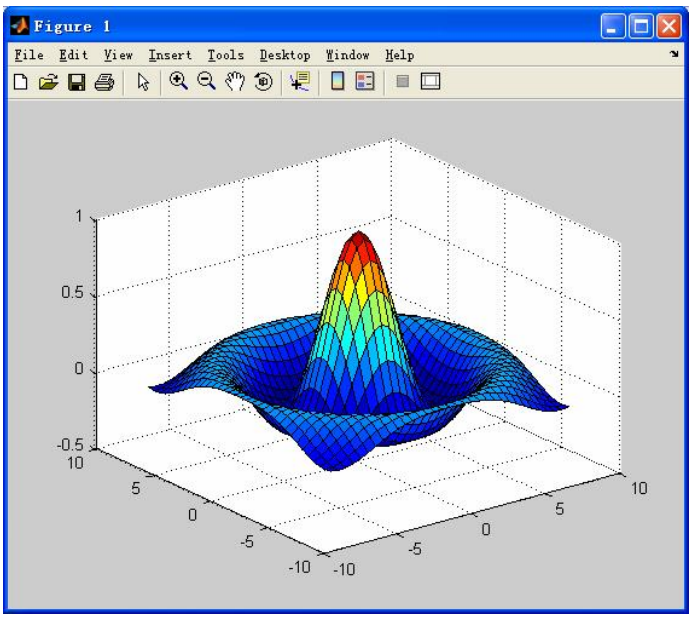
```
>> [X,Y] = meshgrid(-3:.125:3);
>> Z = peaks(X,Y);
>> meshc(X,Y,Z);
>> axis([-3 3 -3 3 -10 5])
```



✓ Surface Plots

Use surf and surfc to view mathematical functions over a rectangular region. surf and surfc create colored parametric surfaces specified by X, Y, and Z, with color specified by Z or C.

```
>> [X,Y] = meshgrid(-8:.5:8);
>> R = sqrt(X.^2 + Y.^2) + eps;      %prevents the divide by zero
>> Z = sin(R)./R;
>> surf(X,Y,Z)
```



✧ **Examples**

✓ Displaying Nonuniform Data on a Surface

This example evaluates the sinc function at random points within a specific range and then generates uniformly sampled data for display as a surface plot. The process involves these tasks:

-Use linspace to generate evenly spaced values over the range of your unevenly sampled data.

-Use meshgrid to generate the plotting grid with the output of linspace.

-Use griddata to interpolate the irregularly sampled data to the regularly spaced grid returned by meshgrid.

-Use a plotting function to display the data. First, generate unevenly sampled data within the range [-8, 8] and use it to evaluate the function.

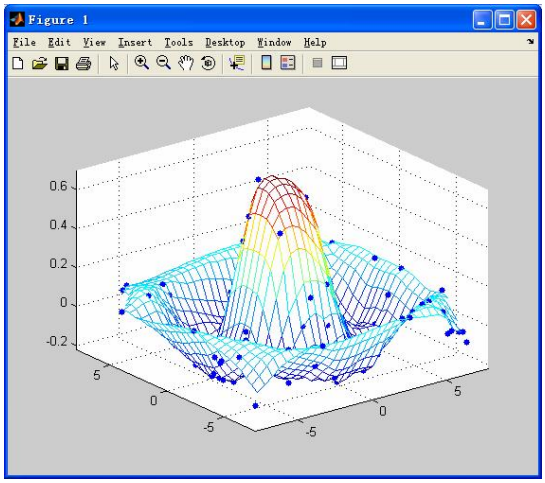
```
>> x = rand(100,1)*16 - 8;
>> y = rand(100,1)*16 - 8;
>> r = sqrt(x.^2 + y.^2) + eps;
>> z = sin(r)./r;
```

The linspace function provides a convenient way to create uniformly spaced data with the desired number of elements. The following statements produce vectors over the range of the random data with the same resolution as that generated by the -8:.5:8 statement in the previous sinc example.

```
>> xlin = linspace(min(x),max(x),33);
>> ylin = linspace(min(y),max(y),33);
>> [X,Y] = meshgrid(xlin,ylin);        %generate a uniformly spaced grid.
```

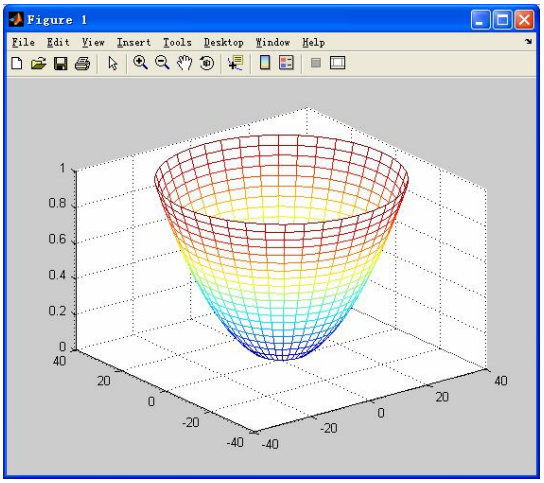
The key to this process is to use griddata to interpolate the values of the function at the uniformly spaced points, based on the values of the function at the original data points (which are random in this example).

```
>> Z = griddata(x,y,z,X,Y,'cubic');      %uses a triangle-based cubic interpolation to …
                                         generate the new data.
>> mesh(X,Y,Z)                           % Plotting the interpolated and the nonuniform …
                                         data produces
>> axis tight; hold on
>> plot3(x,y,z,'.','MarkerSize',15)      %nonuniform
```



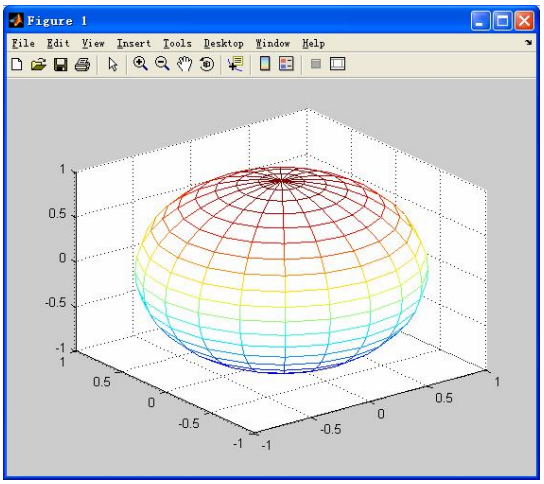
✓ Surface of Rotation

```
>> z=0:20;
>> R=(60*z).^(1/2);
>> [X,Y,Z]=cylinder(R,40);
>> mesh(X,Y,Z)
```



✓ Surface of Sphere

```
>> [X,Y,Z]=sphere;
>> mesh(X,Y,Z)
```



✧ Appendix

RGB Color Components

This table lists some representative RGB color definitions.

Red	Green	Blue	Color
0	0	0	Black
1	1	1	White
1	0	0	Red
0	1	0	Green
0	0	1	Blue
1	1	0	Yellow
1	0	1	Magenta
0	1	1	Cyan
0.5	0.5	0.5	Gray
0.5	0	0	Dark red
1	0.62	0.40	Copper
0.49	1	0.83	Aquamarine