Lab #2 (plot tutorial)

Adam Jaso
09/15/2011

An Introduction to Plotting in MATLAB

Objective:

The objective of this lab is to gain experience in using MATLAB plotting functions.

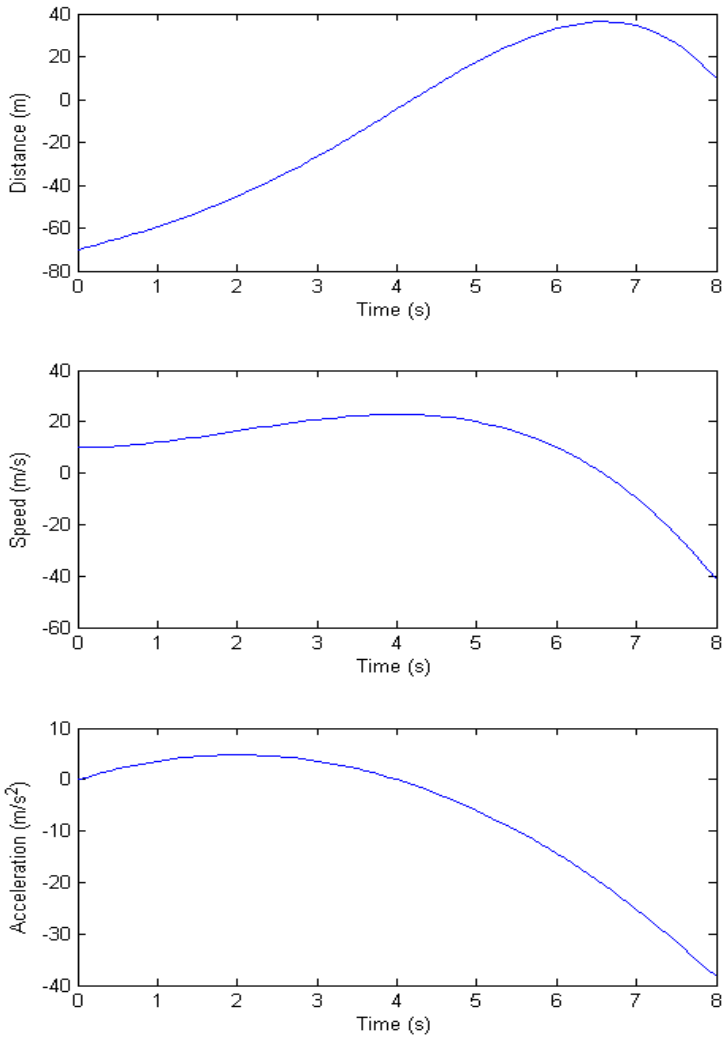
Procedure:

First, we were to complete problems 3 and 4 from Activity #3. Problem 3 required the generation of the position, speed, and acceleration subplots for a given position equation. The attached code and plot demonstrate the solution to this problem. Problem 4 required the creation of a Hertzprung-Russell plot for the theoretical and measured luminosity ratios to temperature ratios function: L / L_sun = (R / R_sun)^2 * (T / T_sun)^4. The attached code and plot demonstrate the solution to this problem. Second, Chapter 9 from Gilat's book was assigned. The procedure was to systematically go through all the examples in the provided pages from this book testing them in MATLAB and altering the different inputs to observe the effects of each input parameter. In addition to this systematic examination of MATLAB plotting features, we were to use these plotting functions to develop an alternate solution to one of the example problems at the end of the chapter. I chose the projectile problem. My solution determines the real three dimensional path of a projectile using the simple kinematic equation: **r = r0** + **v0** * t + 0.5 * **g** * t ^ 2 where bold face indicates a three dimensional vector.

Results / Code:

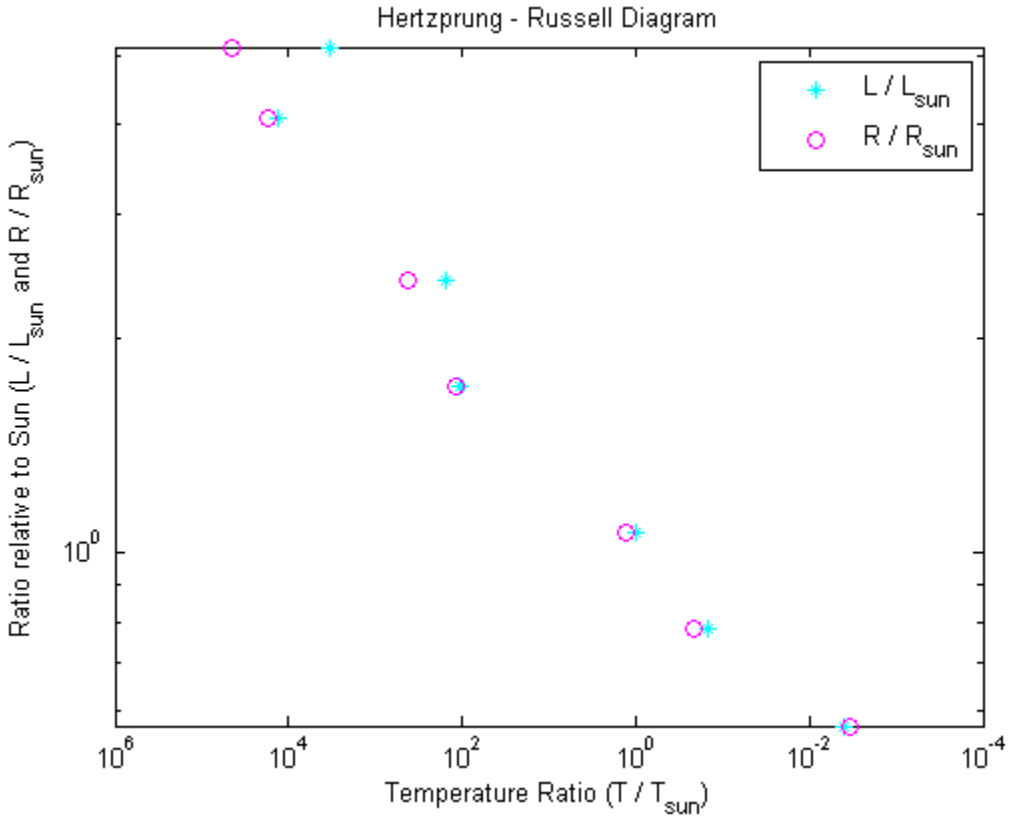
Activity #3 Problem 3 Solution:

```
%act3prob3.m
%This is a solution to Activity
#3 Problem #3

xAxisLabel='Time (s)';
t=0:0.1:8;
x=-.1*t.^4+.8*t.^3+10*t-70;
v=-.4*t.^3+2.4*t.^2+10;
a=-1.2*t.^2+4.8*t;

subplot(3,1,1),plot(t,x);
xlabel(xAxisLabel);
ylabel('Distance (m)');
subplot(3,1,2),plot(t,v);
xlabel(xAxisLabel);
ylabel('Speed (m/s)');
subplot(3,1,3),plot(t,a);
xlabel(xAxisLabel);
ylabel('Acceleration (m/s^2)');
```



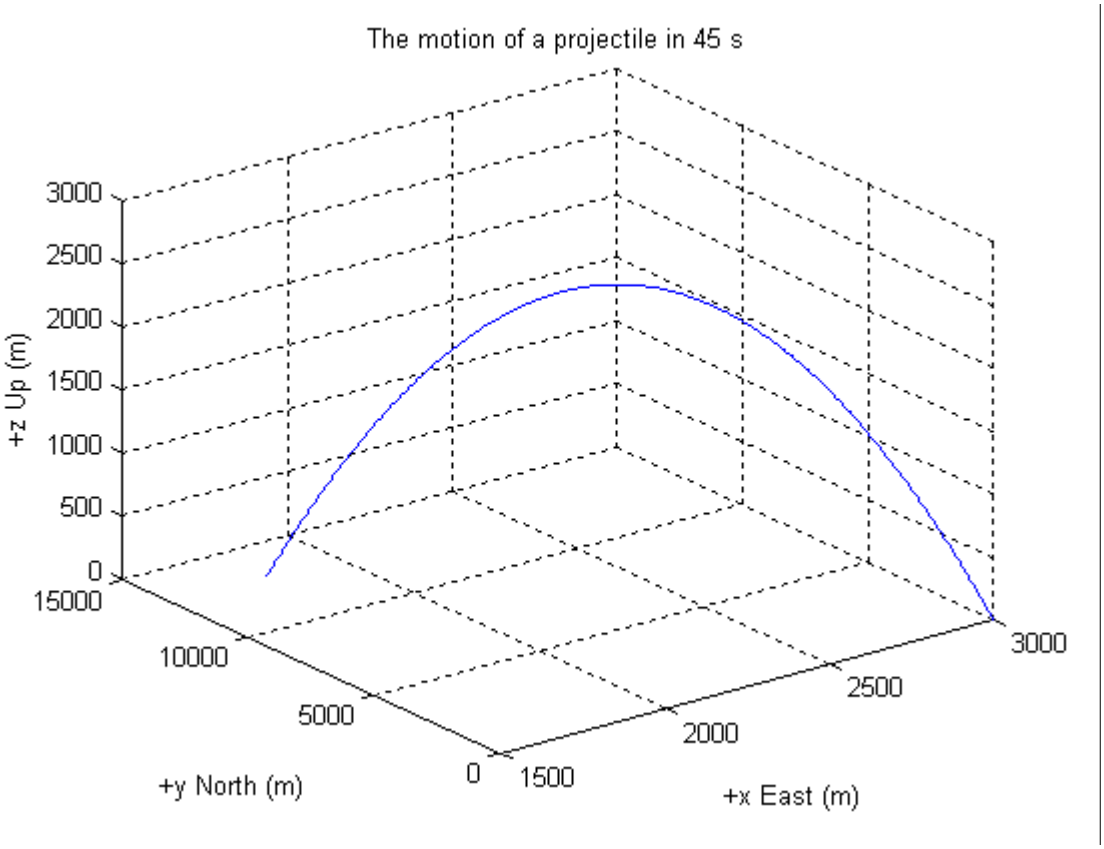
Activity #3 Problem 4 Solution:

```
%act3prob4.m
%This is a solution to Activity #3 Problem 4

T=[5840 22400 13260 9400 3130 4280 28200]/5480;
LLdata=[1 13400 150 108 0.004 0.15 3400];
RRdata=[1 7.8 3.5 3.7 0.18 0.76 8];
LLsun=(RRdata.^2).*(T.^4);
p1=plot(LLdata,T,'c*');
xlabel('Temperature Ratio (T / T_s_u_n)');
ylabel('Ratio relative to Sun (L / L_s_u_n and R / R_s_u_n)');
set(gca,'XScale','log');
set(gca,'YScale','log');
set(gca,'XDir','reverse');
hold on;
p2=plot(LLsun,T,'mo');
set(gca,'YScale','log');
set(gca,'XDir','reverse')
legend('L / L_s_u_n','R / R_s_u_n');
title('Hertzprung - Russell Diagram');
```



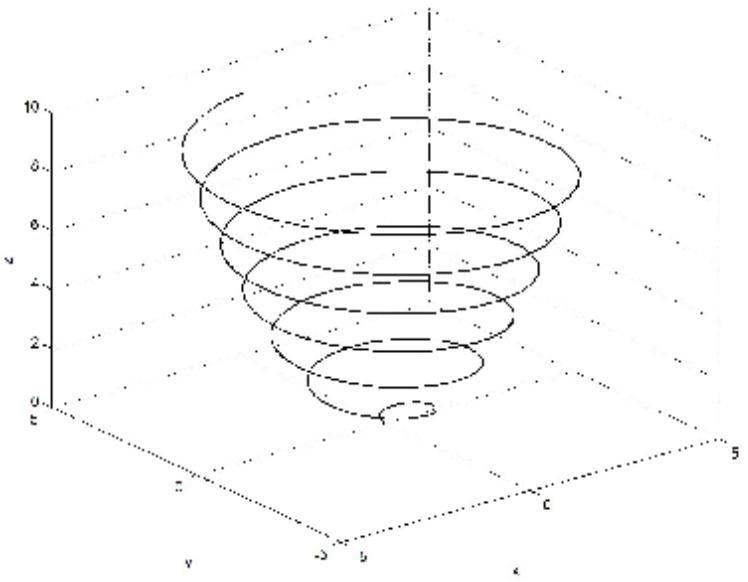
Alternate Solution to Gilat's Chapter 9 Projectile Example:

```
%projectile.m
%This script generates the plot of the path of a fired projectile
moving in
%a frictionless environment above the surface of earth

alt=pi/180*65;
az=pi/180*90;
windv=[-30,0,0];
v0mag=250;
r0=[3000,0,0];
v0=[v0mag*cos(az),v0mag*sin(az),v0mag*sin(alt)]+windv;
g=[0,0,9.81];

t=0:0.1:45;
x=r0(1)+v0(1)*t-.5*g(1)*t.^2;
y=r0(2)+v0(2)*t-.5*g(2)*t.^2;
z=r0(3)+v0(3)*t-.5*g(3)*t.^2;
plot3(x,y,z);
title('The motion of a projectile in 45 s');
xlabel('+x East (m)');
ylabel('+y North (m)');
zlabel('+z Up (m)');
grid on;
```



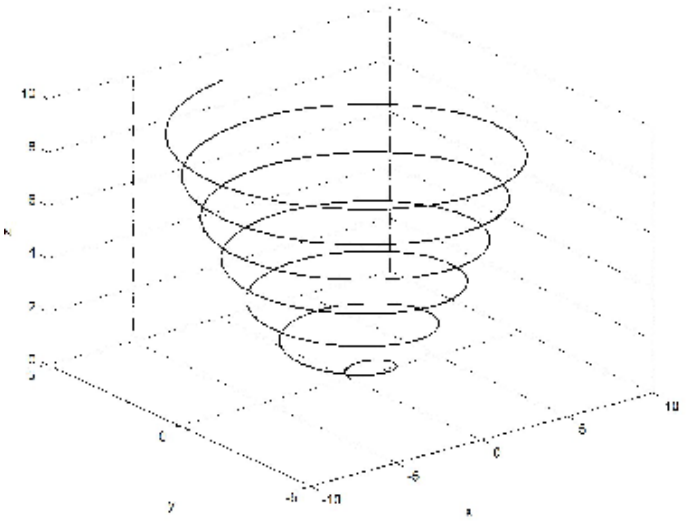
Gilat's Chapter 9 Examples:
Below is shown the MATLAB command window printout with the generated figures added.

```
>> t=0:0.1:6*pi;...
x=sqrt(t).*sin(2*t);...
y=sqrt(t).*cos(2*t);...
z=.5*t;...
plot3(x,y,z,'k','linewidth',1);...
grid on;

>> t=0:0.1:6*pi;...
x=sqrt(t).*sin(2*t);...
y=sqrt(t).*cos(2*t);...
z=.5*t;...
plot3(x,y,z,'k','linewidth',1);...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
```



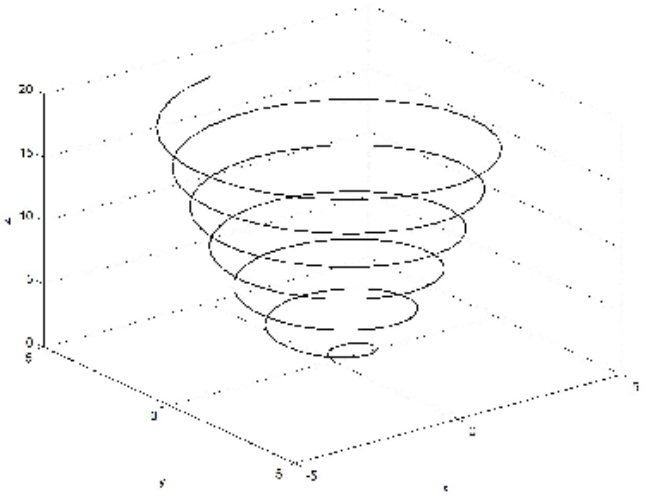
plot3(x,y,z,'k','linewidth',1)

```
>> t=0:0.1:6*pi;...
x=sqrt(t).*sin(2*t);...
y=sqrt(t).*cos(2*t);...
z=.5*t;...
plot3(2*x,y,z,'k','linewidth',1);...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
```



*plot3(2*x,y,z,'k','linewidth',1)*

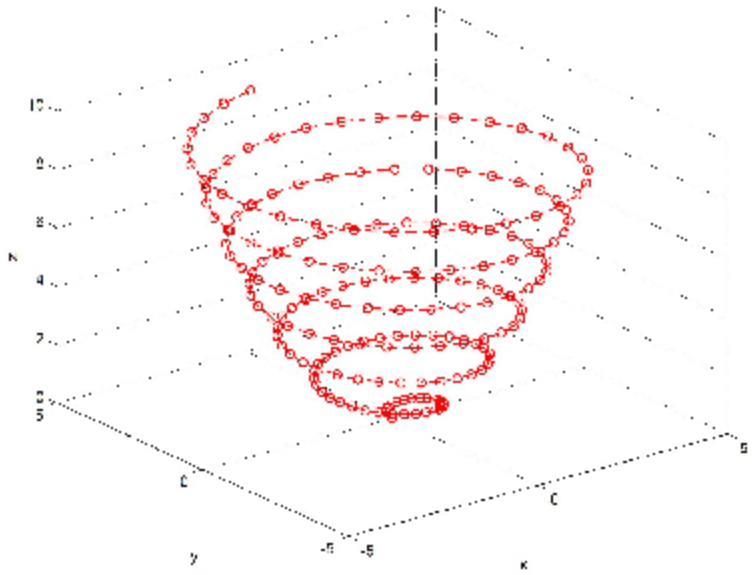
```
>> t=0:0.1:6*pi;...
x=sqrt(t).*sin(2*t);...
y=sqrt(t).*cos(2*t);...
z=.5*t;...
plot3(x,2*y,z,'k','linewidth',1);...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
>>
>> t=0:0.1:6*pi;...
x=sqrt(t).*sin(2*t);...
y=sqrt(t).*cos(2*t);...
z=.5*t;...
plot3(x,y,2*z,'k','linewidth',1);...
```



*plot3(x,y,2*z,'k','linewidth',1)*

```
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
```

```
>> t=0:0.1:6*pi;...
x=sqrt(t).*sin(2*t);...
y=sqrt(t).*cos(2*t);...
z=.5*t;...
plot3(x,y,z,'r--o','linewidth',1);...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
```

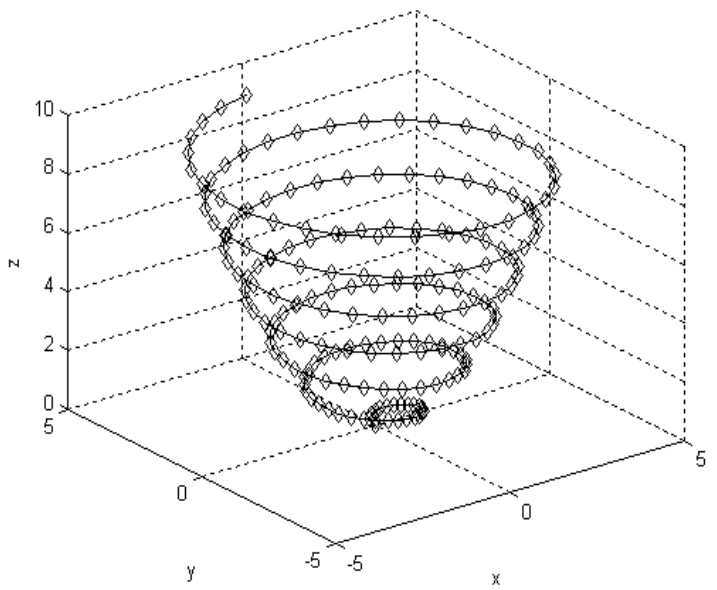


plot3(x,y,z,'r--o','linewidth',1)

```
>> t=0:0.1:6*pi;...
x=sqrt(t).*sin(2*t);...
y=sqrt(t).*cos(2*t);...
z=.5*t;...
plot3(x,y,z,'k','xdatasource','the x data source');...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
```

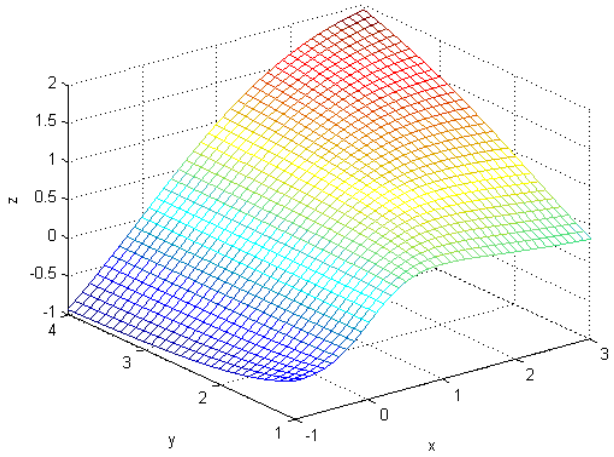
```
>> t=0:0.1:6*pi;...
x=sqrt(t).*sin(2*t);...
y=sqrt(t).*cos(2*t);...
z=.5*t;...
plot3(x,y,z,'k','marker',d);...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
??? Undefined function or variable 'd'.
```

```
>> t=0:0.1:6*pi;...
x=sqrt(t).*sin(2*t);...
y=sqrt(t).*cos(2*t);...
z=.5*t;...
plot3(x,y,z,'k','marker','d');...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
```



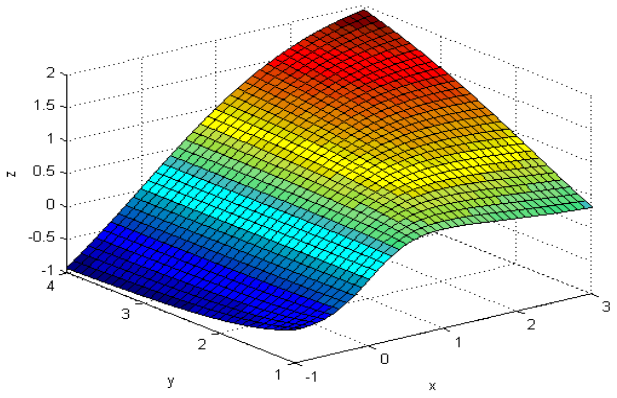
plot3(x,y,z,'k','marker','d')

```
>> 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);...
xlabel('x'); ylabel('y'); zlabel('z');
```



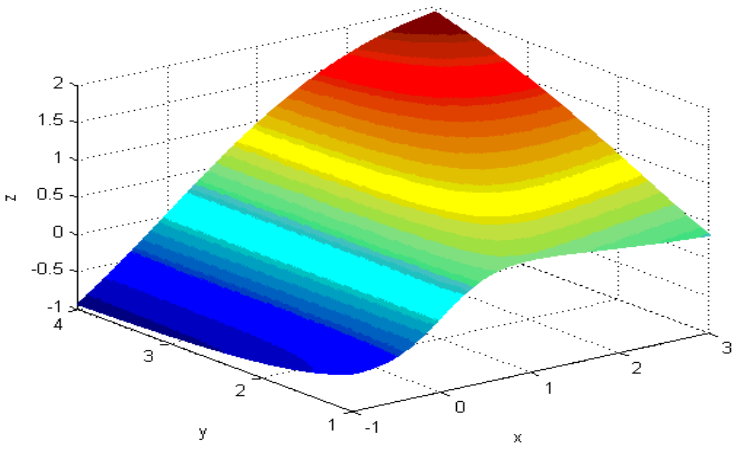
mesh(X,Y,Z)

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



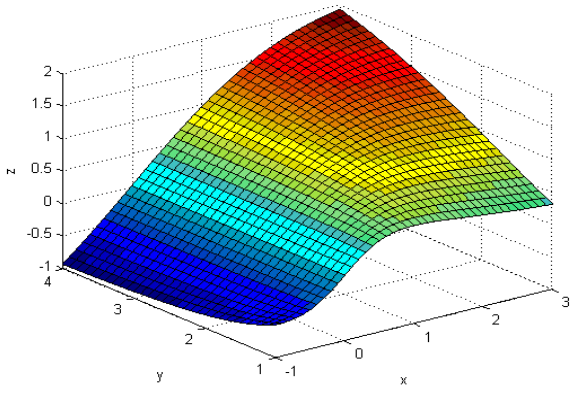
surf(X,Y,Z)

```
>> x=-1:0.01:3;...
y=1:0.01:4;...

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



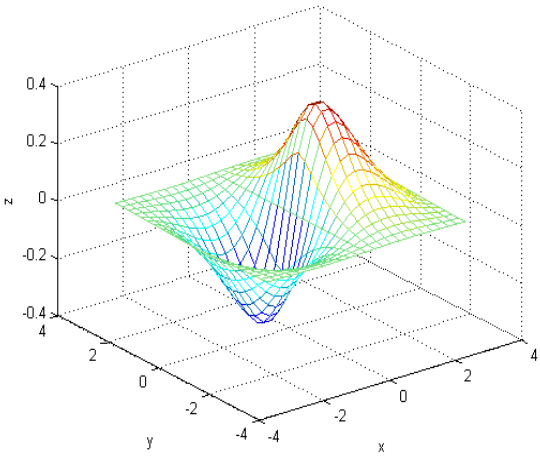
mesh(X,Y,Z)

```
>> x=-1:0.1:3;...

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



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(.5*Y).*sin(X);...
mesh(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
??? Error using ==> plus
Matrix dimensions must agree.
```

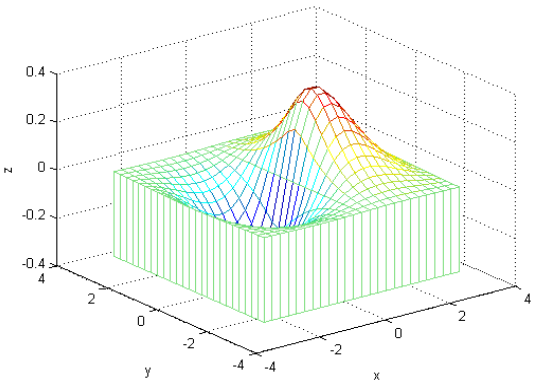


mesh(X,Y,Z)

```
>> 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(.5*Y).*sin(X);...
mesh(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('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(.5*Y).*sin(X);...
surf(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
??? Error using ==> plus
Matrix dimensions must agree.
```

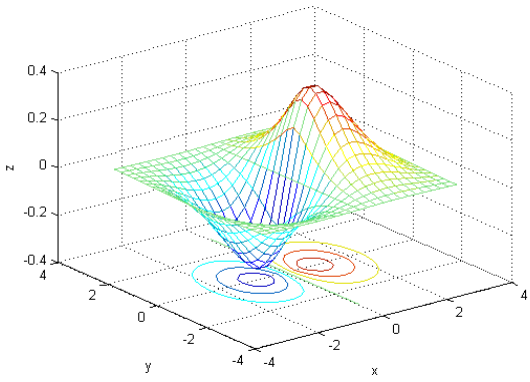
```
>> x=-3:0.25:3;...
y=-3:0.25:3;...
[X,Y]=meshgrid(x,y);...
Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(.5*Y).*sin(X);...
surf(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
```



meshz(X,Y,Z)

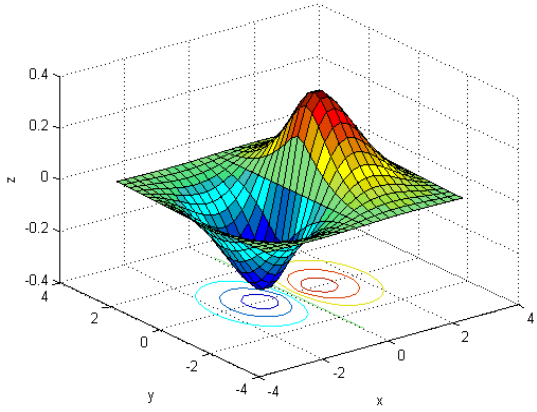
```
>> x=-3:0.25:3;...
y=-3:0.25:3;...
[X,Y]=meshgrid(x,y);...
Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(.5*Y).*sin(X);...
meshz(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
```

```
>> 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(.5*Y).*sin(X);...
meshc(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
```



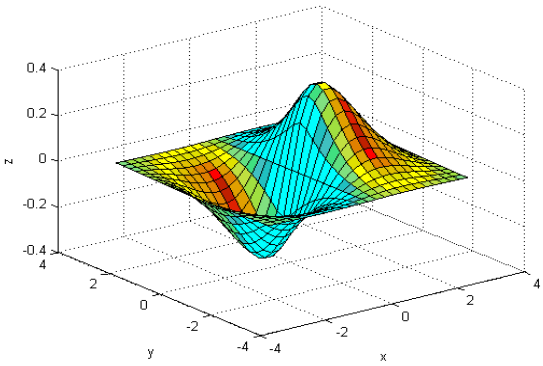
meshc(X,Y,Z)

```
>> 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(.5*Y).*sin(X);...
surfc(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
```



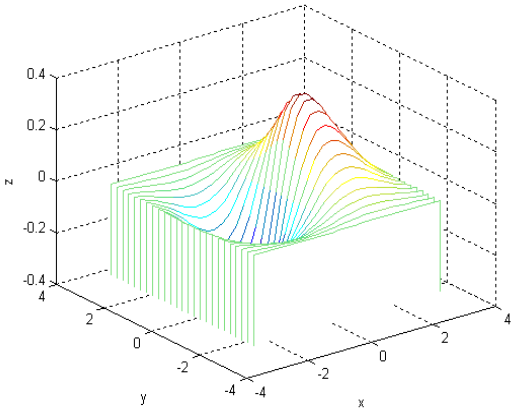
surfc(X,Y,Z)

```
>> 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(.5*Y).*sin(X);...
surfl(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
```



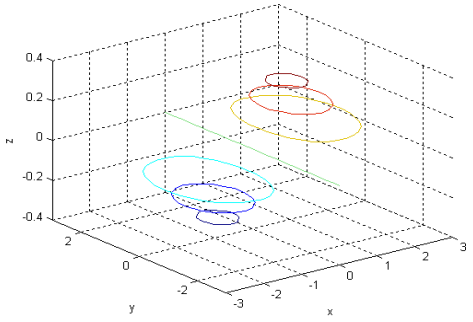
surfl(X,Y,Z)

```
>> 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(.5*Y).*sin(X);...
water-fall(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
??? Undefined function or variable 'water'.

>> 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(.5*Y).*sin(X);...
waterfall(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
```



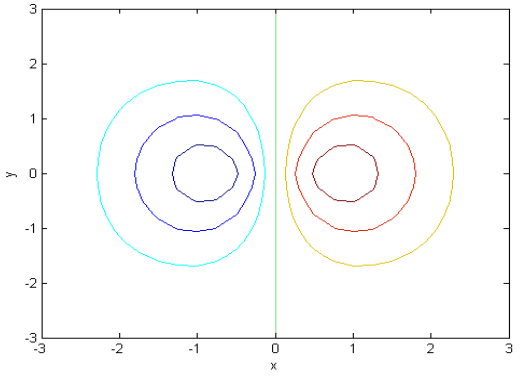
waterfall(X,Y,Z)

```
>> 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(.5*Y).*sin(X);...
contour3(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
```



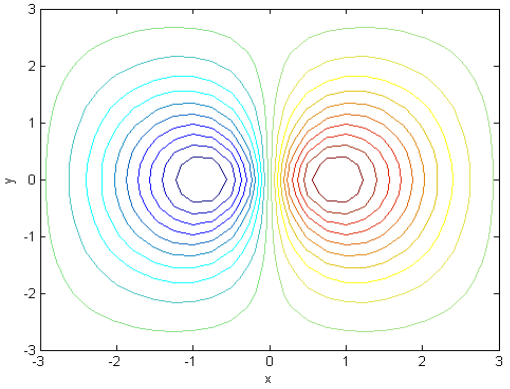
contour3(X,Y,Z)

```
>> 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(.5*Y).*sin(X);...
contour(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
```



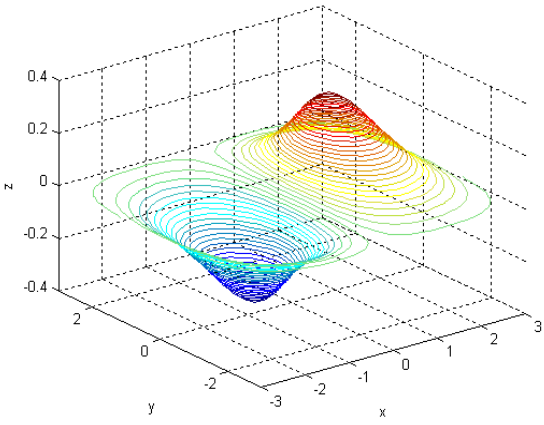
contour(X,Y,Z)

```
>> 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(.5*Y).*sin(X);...
contour(X,Y,Z,20);...
xlabel('x'); ylabel('y'); zlabel('z');
```



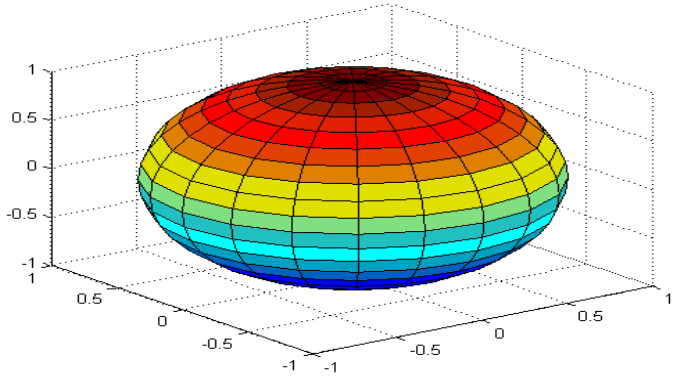
contour(X,Y,Z,20)

```
>> 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(.5*Y).*sin(X);...
contour3(X,Y,Z,50);...
xlabel('x'); ylabel('y'); zlabel('z');
```



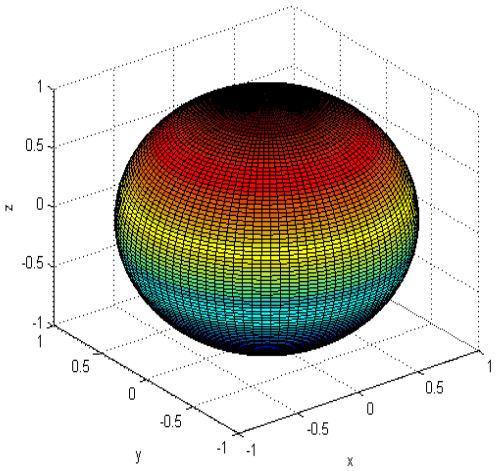
contour3(X,Y,Z,50)

```
>> [X,Y,Z]=sphere(20);
>> [X,Y,Z]=sphere(20);...
surf(X,Y,Z);
```



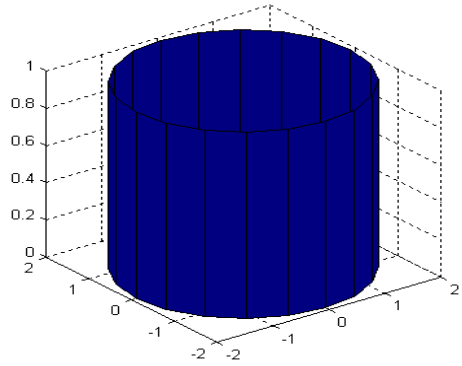
sphere(20)

```
>> [X,Y,Z]=sphere(20);...

surf(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
>> [X,Y,Z]=sphere(10000);...
surf(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
??? Error using ==> mtimes
Out of memory. Type HELP MEMORY for your options.

Error in ==> sphere at 33
x = cosphi*cos(theta);

>> [X,Y,Z]=sphere(10000);...
>> [X,Y,Z]=sphere(1000);...
surf(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
>> [X,Y,Z]=sphere(100);...
surf(X,Y,Z);...
xlabel('x'); ylabel('y'); zlabel('z');
```



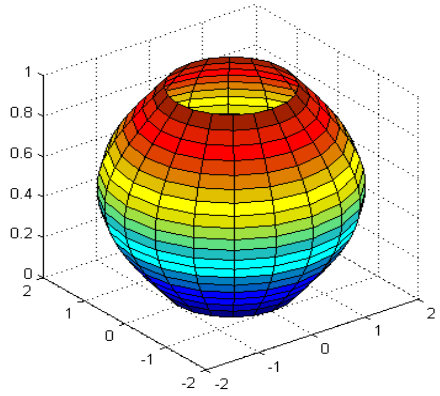
sphere(100)

```
>> t=linspace(0,pi,20);...
r=1+sin(2);...

[X,Y,Z]=cylinder(r);...
surf(X,Y,Z);...
axis square;
```



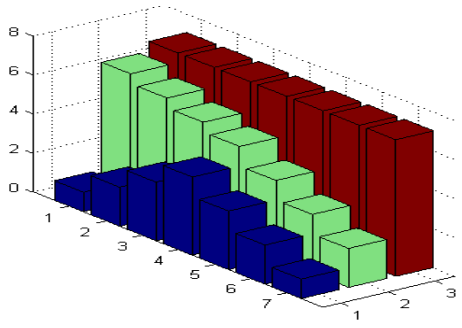
cylinder(1+sin(2))

```
>> t=linspace(0,pi,20);...
r=1+sin(t);...
[X,Y,Z]=cylinder(r,'marker','r');...
surf(X,Y,Z);...
axis square;
??? Error using ==> colon
First and last colon operands must be char.

Error in ==> cylinder at 33
theta = (0:n)/n*2*pi;
>> t=linspace(0,pi,20);...
r=1+sin(t);...
[X,Y,Z]=cylinder(r);...
surf(X,Y,Z);...
axis square;

>> Y=[1 6.5 7;2 6 7; 3 5.5 7; 4 5 7; 3 4 7 2 3
7; 1 2 7];...
bar3(y);
```



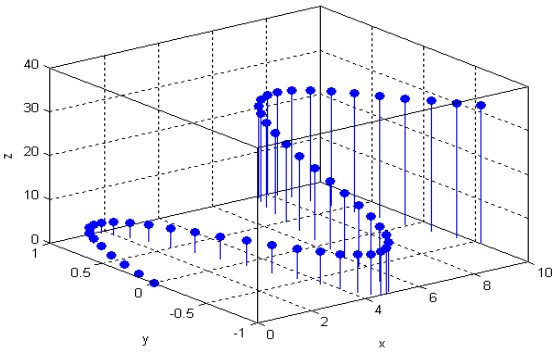
cylinder(1+sin(t))

```
??? Error using ==> vertcat
CAT arguments dimensions are not consistent.
>> Y=[1 6.5 7;2 6 7; 3 5.5 7; 4 5 7; 3 4 7 2 3 7; 1 2 7];...
bar3(Y);
??? Error using ==> vertcat
CAT arguments dimensions are not consistent.
>> Y=[1 6.5 7;2 6 7; 3 5.5 7; 4 5 7; 3 4 7; 2 3 7; 1 2 7];...
bar3(Y);
```



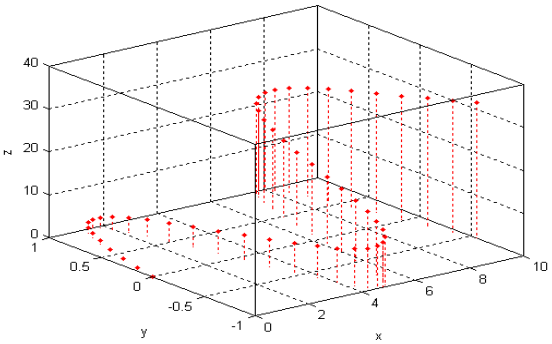
bar3(Y)

```
>> t=0:0.2:10;...
x=t;...
y=sin(t);...
z=t.^1.5;...
stem3(x,y,z,'fill');...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
```



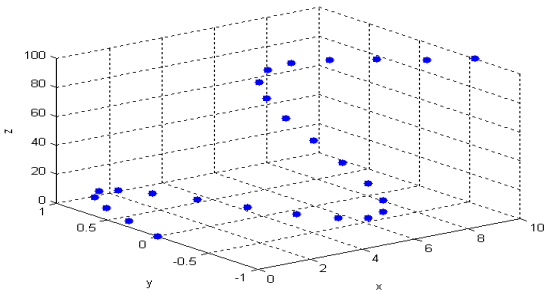
stem3(x,y,z,'fill')

```
>> stem3(x,y,z,'r:.');...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
```



stem3(x,y,z,'r:.')

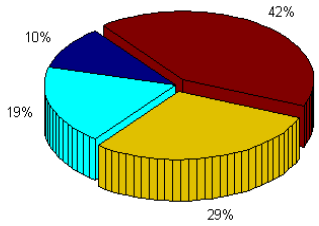
```
>> t=0:0.4:10;...

x=t;...
y=sin(t);...
z=t.^2;
```



scatter3(x,y,z,'filled')

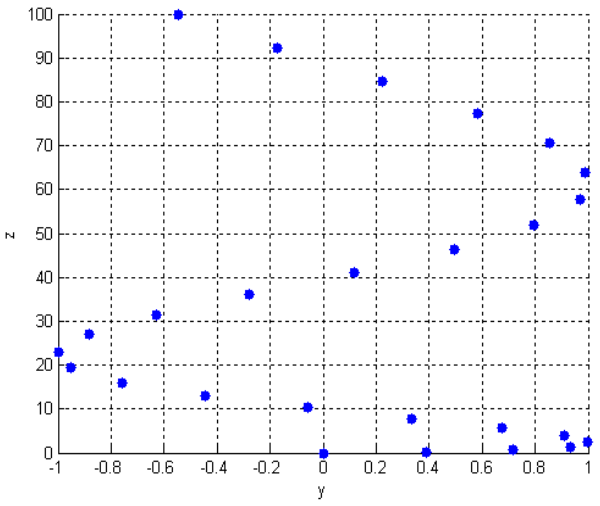
```
>> scatter3(x,y,z,'filled');...
grid on;...
colormap([0.1 0.1 0.1]);...
xlabel('x'); ylabel('y'); zlabel('z');
```

```
>> X=[5 9 14 20];...
explode([0 0 1 1]);...
pie3(X,explode)
??? Undefined function or method 'explode' for input arguments of type
'double'.
>>  X=[5 9 14 20];...
explode=[0 0 1 1];...
pie3(X,explode)
>> colormap(default);
??? Undefined function or variable 'default'.
>> colormap([0.5 0.5 0.5]);
>>  X=[5 9 14 20];...
explode([0 0 1 1]);...
pie3(X,explode)
??? Subscript indices must either be real positive integers or logicals.

>>  X=[5 9 14 20];...
explode=[0 0 1 1];...
pie3(X,explode)
```



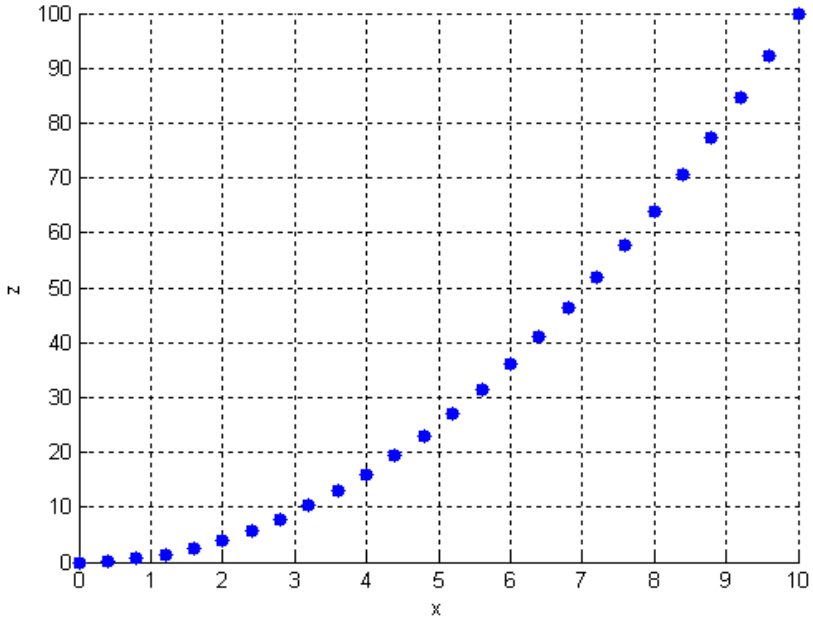
pie3(X,explode)

```
>> t=0:0.4:10;...
x=t;...
y=sin(t);...

z=t.^2;
>> scatter3(x,y,z,'filled');...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
>> scatter3(x,y,z,'filled');...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');
>> scatter3(x,y,z,'filled');...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');...
view(90,0);
```



scatter3(x,y,z,'filled');view(90,0);

```
>> scatter3(x,y,z,'filled');...
grid on;...
xlabel('x'); ylabel('y'); zlabel('z');...

view(0,0);
>>
```



scatter3(x,y,z,'filled');view(0,0);

Observations:

While all of these plots supply the modeling different needs of many different experiments and data sets, I believe that the scatter3() plot and the mesh() plot are the most universally useful. The ability with the scatter3 plot to visualize actual data points in space and to make visual connections without suggested lines is extremely helpful. The mesh plot also provides a transparent view of a surface from data or from a theoretical model.