Armstrong Atlantic State University
Engineering Studies
MATLAB Marina – 3D Plotting Primer

Prerequisites
The 3D Plotting Primer assumes knowledge of the MATLAB IDE, MATLAB help, arithmetic operations, built in functions, scripts, variables, arrays, logic expressions, conditional structures, iteration, functions, debugging, characters and strings, cell arrays, structures, file input and output, and 2D plotting. Material on these topics is covered in the MATLAB Marina Introduction to MATLAB module, MATLAB Marina Variables module, MATLAB Marina Arrays module, MATLAB Marina Logic Expressions module, MATLAB Marina Conditional Structures module, MATLAB Marina Iteration module, MATLAB Marina Functions module, MATLAB Marina debugging module, MATLAB Marina Character and Strings module, MATLAB Marina Cell Arrays module, MATLAB Marina Structures module, MATLAB Marina File Input and Output module, and MATLAB Marina Plotting module.

Learning Objectives
1. Be able to generate and appropriately annotate linear 3D plots using MATLAB.
2. Be able to generate and appropriately annotate surface and contour plots using MATLAB.
3. Be able to determine the appropriate type of plot to present scientific/engineering data.

Terms
figure window, 2D plot, linear 3D plot, parametric plot, surface plot, contour plot

MATLAB Functions, Keywords, and Operators
figure, xlabel, ylabel, zlabel, title, legend, gtext, grid, axis, clf, close, plot3, view, meshgrid, surf, mesh, contour

Linear 3D Plots
Linear parametric 3D plots are plots where the x, y, and z values are functions of an independent variable or variables often time or position. Consider 3D projectile motion, the x-position, y-position, and z-position of the projectile are all functions of the independent variable time. The MATLAB function `plot3` can be used for linear parametric 3D plots.

Recall from Physics the kinematic equations for motion in three dimensions are
$v_x = v_{x0} + a_x t$, $v_y = v_{y0} + a_y t$, $v_z = v_{z0} + a_z t$ velocity as a function of time
$s_x = s_{x0} + v_{x0} t + \frac{1}{2} a_x t^2$ displacement as a function of time
$s_y = s_{y0} + v_{y0} t + \frac{1}{2} a_y t^2$
$s_z = s_{z0} + v_{z0} t + \frac{1}{2} a_z t^2$
Where s is position (s_0 is the initial position), v is velocity (v_0 is the initial velocity), a is acceleration, and t is time. The subscripts x, y, and z indicate the direction for each of the

components. Figure 1a shows the linear parametric 3D plot of the projectile motion and Figure 1b shows the MATLAB program used to generate the plot of projectile motion.

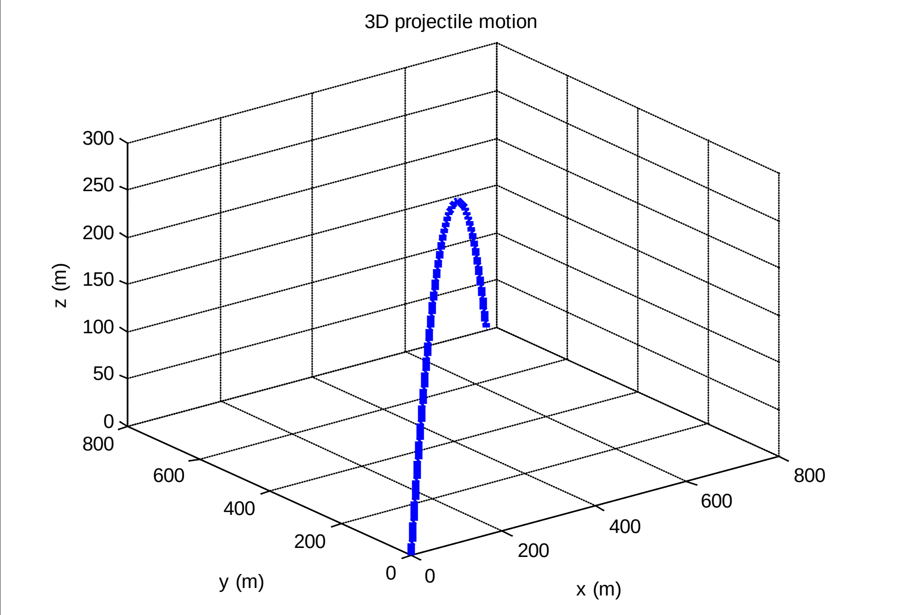

Figure 1a, 3D Projectile Motion

MATLAB allows one to specify the viewing angle of plots. The viewing angle can be specified for both 2D and 3D plots. The MATLAB function `view(azimuth, elevation)` allows one to specify the angle and elevation one sees the plot from. The azimuth is horizontal rotation in degrees (about the z-axis) and the elevation is the vertical elevation in degrees. Positive azimuth angles correspond to rotation counter-clockwise about the z-axis. Positive elevation values correspond to viewing from above and negative values of elevation viewing from below.

Some common azimuths and elevations are:

- azimuth = 0, elevation = 90 is directly overhead and the default 2-D view.
- azimuth = elevation = 0 looks directly up the first column of the matrix.
- azimuth = 180 is behind the matrix.
- view(2) sets the default 2-D view, azimuth = 0, elevation = 90.
- view(3) sets the default 3-D view, azimuth = -37.5, elevation = 30.

The viewing angle can also be set from the axis properties of the figure window: `Edit – Axes Properties`. From the Axes Properties choose More Properties and the viewing angle can be altered by changing the view property. The rotate tool (counter-clockwise circular arrow) on the figure window toolbar can also be used to change the viewing angle.

```
% 3D parametric plotting (projectile motion in 3D
% assume initial position is x = y = z = 0
sx0 = 0.0;
sy0 = 0.0;
sz0 = 0.0;
% initial velocities for projectile fired at angle 45 degrees
% with respect to both xy and yz planes
vx0 = 100*cos(pi/4)*sin(pi/4);
vy0 = 100*sin(pi/4)*sin(pi/4);
vz0 = 100*cos(pi/4);
% acceleration due to gravity
ax = 0.0;
ay = 0.0;
az = -9.8;  % gravity in m/s^2

% calculate the projectile motion for t = 0 to 14 seconds
t = 0.0 : 0.01 : 14.0;
sx = sx0 + vx0*t + 0.5*ax*t.*t;
sy = sy0 + vy0*t + 0.5*ay*t.*t;
sz = sz0 + vz0*t + 0.5*az*t.*t;

% parametric 3D plot of position
figure(1)
plot3(sx,sy,sz,'Linewidth',4);
grid
xlabel('x (m)');
ylabel('y (m)');
zlabel('z (m)');
title('3D projectile motion');
```

Figure 1b, MATLAB Program to Plot 3D Projectile Motion

3D Surface Plots

Surface plots are used to plot a variable that is a function of two independent variables. The two independent variables specify a 2D grid which the third variable is plotted versus.

The process for creating surface plots consists of three steps:

1. Create a grid of points specifying all the valid points in the x-y plane (all x y locations). Typically the `meshgrid` function is used for this.
2. Compute the z values for the grid.
3. Plot the surface plot using one of the MATLAB 3D surface plot functions: `mesh, meshc, surf,` or `surfc.`

Consider plotting the paraboloid $z = 4 - x^2 - y^2$ over the range $-5 \le x \le 5$ and $-5 \le y \le 5$. The MATLAB program of Figure 2a generates the paraboloid of Figure 2b.

```
x = linspace(-5.0,5.0,100);
y = linspace(-5.0,5.0,100);
[xx, yy] = meshgrid(x,y);
zz = 4 - xx.^2 - yy.^2;

figure(1)
mesh(xx,yy,zz);
xlabel('x');
ylabel('y');
zlabel('z');
title('Paraboloid');
```

Figure 2a, MATLAB Program for Plot of Paraboloid

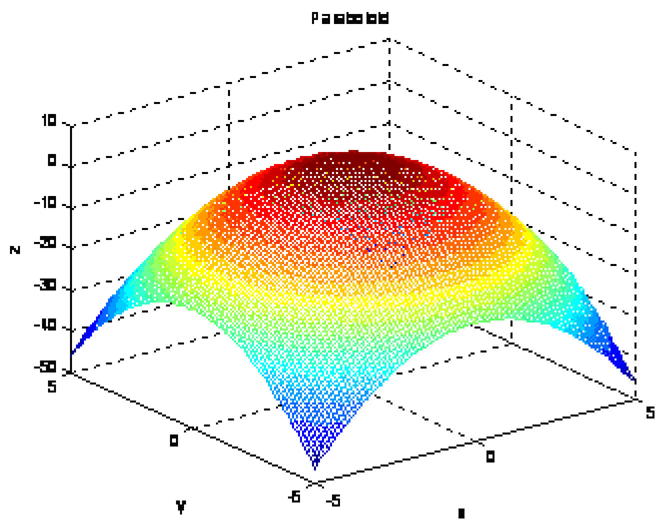

Figure 2b, Surface Plot of Paraboloid

The `meshgrid` function takes two vectors specifying the edges of the grid for the 3D plot and replicates the rows and columns to produce rectangular matrices specifying the underlying grid for a 3D plot. Figure 3 illustrates the result of the `meshgrid` function.

The `mesh` function generates a surface plot with the surfaces as white facets outlined by colored lines. The line colors are proportional to the z axis values. The `surf` function generates a surface plot with the surfaces as colored facets outlined by black lines. The surface colors are proportional to the z axis values. The functions `meshc` and `surfc` generate mesh and surface plots just as mesh and surf do but also generate a contour plot below the surface plot.

For 3D surface plots, the xyz coordinates specify points in three space and the 3D surface plot functions generate smooth surfaces connecting adjacent points.

```
>> x=[1:1:3]
x = 1  2  3
>> y=2:2:6
y = 2  4  6
>> [xx,yy] = meshgrid(x,y)
xx =
     1     2     3
     1     2     3
     1     2     3
yy =
     2     2     2
     4     4     4
     6     6     6
```

Figure 3, `meshgrid` Function Results

The paraboloid of Figure 2b was generated using rectangular coordinates. Notice that the chosen range does not give all the z = 0 points. Some surface plots are easier to specify using polar (r,θ,z) or spherical (ρ,θ,φ) coordinates. To plot the paraboloid for $z \geq 0$, it is easier to specify the points using polar coordinates. The projection of the paraboloid on the x-y plane is a circle of radius 2, $x^2 + y^2 = 4$. One can think of the paraboloid as a series of circles with different radius piled on top of each other, the radius of the circle for a particular z will be $\sqrt{4-z}$. The height of the paraboloid above the x-y plane is 4. The MATLAB program of Figure 3a generates the paraboloid of Figure 3b using polar coordinates for the underlying x-y grid.

```
theta = linspace(0,2*pi,100);
z = linspace(0.0,4.0,100);
[zz,tt] = meshgrid(z, theta);
rr = sqrt(4 - zz);
xx= rr.*cos(tt);
yy = rr.*sin(tt);

figure(1)
mesh(xx,yy,zz)
xlabel('x');
ylabel('y');
zlabel('z');
title('Paraboloid');
```

Figure 3a, MATLAB Program for Plot of Paraboloid

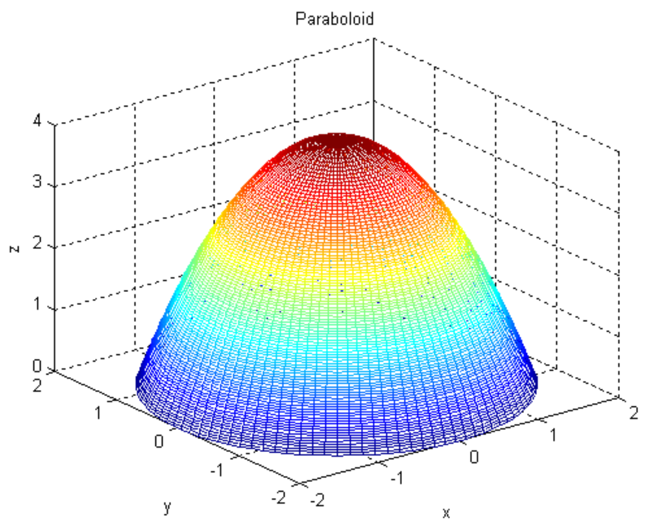

Figure 3b, Surface Plot of Paraboloid

Other useful MATLAB functions for 3D plots are: `shading`, `colormap`, `hidden`, and `lightangle`. See MATLAB's help for more information on these functions.

Contour Plots

MATLAB has functions for 2D and 3D contour plotting: `contour` and `contour3`. For example, the MATLAB statements of Figure 4a generates the 2D and 3D contour plots of the paraboloid $z = 4 - x^2 - y^2$ shown in Figures 4b and 4c.

```
>> figure
>> contour(xx,yy,zz)
>> figure
>> contour3(xx,yy,zz)
```

Figure 4a, MATLAB Statements to Generate 2D and 3d Contour Plots of Paraboloid

The number of contours is determined automatically or can be specified with an optional fourth argument, `contour(xx,yy,zz,N)`or `contour3(xx,yy,zz,N)`where N specifies the number of contours.

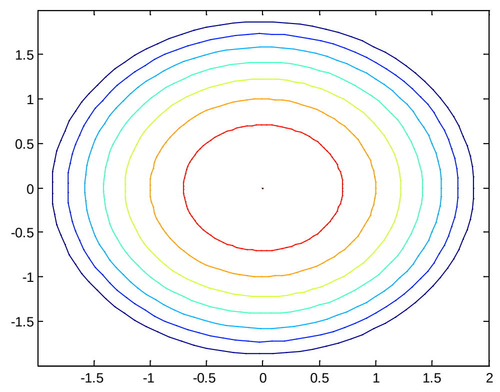

Figure 4b, 2D Contour Plot of Paraboloid

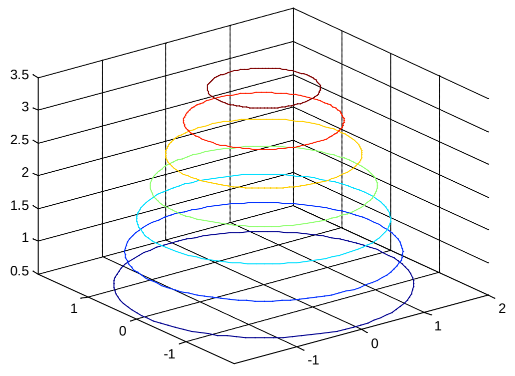

Figure 4c, 3D Contour Plot of Paraboloid

Last modified Monday, September 30, 2013