Basic Surface Plotting in MatLab

Making 3D surface plots, contour plots, and gradient plots in MatLab is slightly more complicated than making simple line graphs, but we will present some examples that, with simple modifications, should enable you to create most of the pictures that you will need.

The first step is to set up vectors that represent the range of x and y values. Suppose we want to plot over the region $-4 \leq x \leq 4$ and $-3 \leq y \leq 3$. Plotting on a grid of about 50 x 50 should be quite adequate, so we set up these vectors as follows:

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

We have to make a *grid* of points over which we want the heights of the surface. MatLab provides a command for this:

```
[X,Y] = meshgrid(x,y);
```

(X and Y will hold the rows and columns of this grid; we could have used any letters for these variables.) Note that we have placed a semicolon (;) after each command; this suppresses the long list of components which MatLab generates as it creates these vectors.

Now suppose that we want to plot the function $z = x^2 - 2xy + 3y + 2$. We create Z from the variables X and Y as follows:

```
Z = X.∧2 - 2*(X.*Y) + 3*Y + 2
```

Now we are ready to plot. The simplest way of getting a surface plot is the MatLab command `surf(X,Y,Z)`. This gives a fixed view with standard colors. There are other palettes of colors available; they are chosen using the `colormap` command. A good all-purpose choice is `colormap(jet)`, which is also good for contour plots. Other color schemes are: `hsv, hot, cool, pink, gray, bone, copper, prism`, and `flag`. It is also possible to rotate the plot in 3D to get different views. In order to do this, issue the command `rotate3d on` before doing any plotting. Now, once you plot the surface, you can rotate it by dragging with the mouse: try it.

You can also choose how your axes are drawn. Estimating the values of the function over the grid of x and y values shows that its values lie roughly between -15 and 40, so we set up our axes as follows: `axis([-4 4 -3 3 -15 40])`. Note that we don't need commas, but we have to use brackets and parentheses: ([ ]).

Once you issue a drawing command such as `surf(X,Y,Z)`, a window containing the plot will be created. To get back to the MatLab command line to adjust the plot (without deleting it), either collapse the window without closing it (click on the "-" in Windows95 e.g.) or click on whatever part of the MatLab worksheet is visible outside the window. It also helps to add semi-colons to each plotting command, so that the window is not reopened each time you add to the plot.

Here then is the sequence of commands we used:

```
≫ x = -4:.1:4; y = -3:.1:3;
≫ [X,Y] = meshgrid(x,y);
≫ Z = X.∧2 - 2*(X.*Y) + 3*Y + 2;
≫ colormap(jet);
≫ surf(X,Y,Z);
≫ axis([-4 4 -3 3 -15 40]);
```

Surfaces and Contours

There are several variations on surface plotting which you may want to experiment with. Instead of `surf(X,Y,Z)`, try `surfl(X,Y,Z), shading interp` or `mesh(X,Y,Z)` or `meshc(X,Y,Z)` (which adds a contour plot).

You can plot contour curves for the function $z = x^2 - 2xy + 3y + 2$ created in the previous section. Instead of the `surf` command, use MatLab's contour drawing command; for example: `contour(X,Y,Z,30)`. This will draw a contour plot on the screen, with 30 different contour levels. They will be color coded, using the same colors as the surface plot.

You can also make a 3D contour plot using `contour3(X,Y,Z)`.

It is sometimes very effective to draw *both* a surface and a contour plot at the same time. To accomplish this, we have to tell MatLab that we want both plots in the same window. This is accomplished using the `hold on` command. Until the `hold off` command is issued (or the graphing window closed), all plots drawn will be superimposed, one on the other. We also have to "raise" the surface plot a bit so that it doesn't get in the way of the contour plot, which is always drawn in the x, y-plane ($z = 0$). We do this by simply adding some convenient number to Z when plotting. The sequence which does the trick is the following:

```
≫surf(X,Y,Z+5);
≫hold on;
≫contour(X,Y,Z+5,30);
```

Using M-files

As you can see, there are lots of parameters that one may want to set in plotting surfaces. Also, the equations themselves take a while to type in. The secret to saving time and being able to make adjustments easily is to use m-files, both for defining the function and for scripting the commands. Here is a file defining the function we have been using.

`func1.m`

```
function z = func1(x,y);
f='x∧2 - 2*(x*y) + 3*y + 2';
z=eval(vectorize(f));
```

Since we have called the function `func1`, we have to name this file `func1.m` so that MatLab can find it when we use it. It is also worth taking some time to see how this definition works. We first define `f` to be a certain series of symbols, enclosed in single quotes: ' '. This is the only part you have to change if you want to change how this function is computed. The command `vectorize` simply adds the correct dots to make this a legal vector definition; for example, `vectorize` changes the `x*y` to `x.*y`. The `eval` command orders MatLab to treat this as a function to be computed, rather than an array of symbols.

Now we can write a script which creates the plot. Here is a sample which draws the surface and the contour plot.

`example1.m`

```
colormap(jet); % Set colors
x=-2:.1:2; y=x; % Set up x and y as vectors
[X,Y]=meshgrid(x,y); % Form the grid for plotting
Z=func1(X,Y); % Create the heights
rotate3d on; % Activate interactive mouse rotation
surf(X,Y,Z+10), shading interp; % Draw surface
hold on; % Allow for more without erasing
axis([-2 2 -2 2 0 25]); % Set up axes
xlabel('x');ylabel('y'); % Label axes
contour3(X,Y,Z+10,30); % 3D contour plot over surface
contour(X,Y,Z+10,30); % 2D contour plot in x,y-plane
```

Adding 10 to the Z-values to show more of the contours was discovered by experiment – another reason for using m-files. The Z-values for the `axis` command, 0 to 25, we found by plotting without this command and seeing how MatLab itself scaled the axes. Since the `contour3` and the `surf` command use the same colors in the same places, you can't see the 3D contours; however, they will appear in a black and white printout.

As a final example, this file produces a plot showing that the spiral helix, $x = t\sin(6x)$, $y = t\cos(6x)$, $z = t/3$, lies on the cone $z = \frac{1}{3}\sqrt{x^2 + y^2}$.

`spiral.m`

```
x=-pi:pi/20:pi; y=x
[X,Y]=meshgrid(x,y);
Z=sqrt(X.∧2 + Y.∧2);
mesh(X,Y,Z/3);
hold on
t=0:pi/200:pi;
h=plot3(t.*sin(6*t),t.*cos(6*t),t/3);
set(h,'linewidth',3);
```

Compare the `mesh` drawing with the `surf` drawing: the first draws a grid or net, while the second fills in the surface. `Plot3` graphs curves in space by giving their x, y and z coordinates in terms of a parameter – in this case, `t`. Setting `h = plot3( )` enables MatLab to change the properties of this particular plot using the `set` command.

Building Peaks and Pits

The family of curves $y = e^{-kx^2}$ looks like a series of "bumps" of height 1, all centered at $x = 0$. Use your calculator or MatLab to plot a few, for $k = 1, 2, 3, 4$ and $-2 \le x \le 2$, on the same set of axes. The curves in the family are similar, but the bigger k, the steeper the curve. We can center the bumps around the point $x = a$ and make them of height c by using $y = ce^{-k(x-a)^2}$ instead. Here is an m-file called `mt.m` which creates three dimensional hills in the same way:

`mt.m`

```
% mt computes heights of a peak centered at (a,b)
% The maximum height is c (which can be negative)
% It uses a function of type exp(-kr∧2); large
% values of k make slimmer peaks
% z = mt(x,y) is the height over the point (x,y)

function z = mt(a,b,c,k,x,y);
f='c*exp(-k*((x-a)∧2+(y-b)∧2))';
z=eval(vectorize(f));
```

If we create vectors X and Y as before, then we can create a single peak at $(1, 1)$ of height 3 with steepness 4, by writing `Z1=mt(1,1,3,4,X,Y)` and plotting `surf(X,Y,Z1)`. We can create a "pit" (opposite of a peak) at $(0, -1)$ of depth 3 by writing `Z2=mt(0,-1,-3,4,X,Y)`. We can plot both with the command `surf(X,Y,Z1+Z2)`.

Exercises: MatLab 3D Plots

1. Create and print out a surface plot and a contour plot for the function $z = (x^2 + 2y^2)e^{1-x^2-y^2}$ over the rectangle $-2 \le x, y \le 2$. You may have to rotate the surface plot

a bit to get a good view. The contour plot should be separate, but you may want to plot contours on the same axes as the surface plot as well. Credit will be based on the quality of the plots. Find out how to label them, or label them by hand.

2. Using the `mt(a,b,c,k,X,Y)` function described above, you can piece together surfaces from peaks and pits. Unfortunately, these mountains and valleys interfere with each other, so that when we add several together, their heights and centers aren't exactly where you expect them to be. This effect can be reduced by making k large, so that the height or depth of a peak or pit approaches 0 very quickly as you move away from its center (a, b).

a. Demonstrate this interference and its improvement by creating some contour plots of two nearby peaks/pits with a few different values of k (e.g. $k = 0.5, k = 2, k = 7$).

b. Create a surface which has a peak of height 2 near $(1, 1)$, another of height 3 near $(1, 1/2)$, and a pit of depth -3.5 near $(-1, 1)$. Print out a good view and a contour diagram (or both in one plot if you want).

3. The surfaces $z = x^2 - y^2$ and $z = xy$ are called "saddles" (centered at $(0, 0)$). Draw a saddle and, separately, its contour plot.

4. Plot $z = 4500 - 105x^2 - 105y^2 + 3y^2x + 3x^2y + 0.8x^4 + 0.8y^4$ over the square $-10 \leq x, y \leq 10$ (use `x=-10:.1:10; y=x`). How many peaks and pits do you see? What about saddle-shaped pieces? Print out a contour plot and label peaks, pits, saddles. Zoom in on a saddle (by changing the axes or resizing x and y) and see if you get a contour diagram resembling the one in the previous exercise.

5. (Optional) Plot an interesting surface of your choosing. Here's one of our favorites:

$$z = \cos(x+y)\cos(3x-y) + \cos(x-y)\sin(x+3y) + 5e^{-(x^2+y^2)/8}$$