

Tutorial 4 - Graphing Surfaces and Vector Functions in Matlab

Nick Rogers

February 10, 2010

Some New Functions

The Mesh Grid command with an Example

The Subplot Command with an Example

Graphing 3d Surfaces

Example - Graphing $x^2 + y^2$

Level Curves

Level Surfaces

Vector Functions

The Gradient Function

Divergence

Curl

meshgrid

"...transforms the domain specified by vectors x and y into arrays X and Y , which can be used to evaluate functions of two variables and three-dimensional mesh/surface plots. The rows of the output array X are copies of the vector x ; columns of the output array Y are copies of the vector y ."

meshgrid

"...transforms the domain specified by vectors x and y into arrays X and Y , which can be used to evaluate functions of two variables and three-dimensional mesh/surface plots. The rows of the output array X are copies of the vector x ; columns of the output array Y are copies of the vector y ."

$[X, Y] = \text{meshgrid}(x, y)$

meshgrid

"...transforms the domain specified by vectors x and y into arrays X and Y , which can be used to evaluate functions of two variables and three-dimensional mesh/surface plots. The rows of the output array X are copies of the vector x ; columns of the output array Y are copies of the vector y ."

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

```
[X,Y] = meshgrid(1:3,10:14)
```

X =

```

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

Y =

```

10    10    10
11    11    11
12    12    12
13    13    13
14    14    14
```

subplot

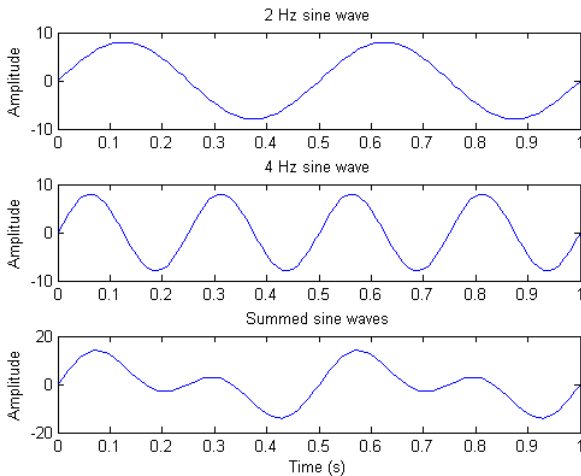
The *subplot* command partitions a figure into subplots. `subplot(m,n,p)` divides the figure into m rows and n columns. The p -value tells matlab which subplot we are talking about. The counting starts top-left, and moves to bottom right.

```
figure
subplot(3,1,1) % top figure
plot(t, s_1)
title('2 Hz sine wave')

subplot(3,1,2) % middle figure
plot(t, s_2)
title('4 Hz sine wave')

subplot(3,1,3) % bottom figure
plot(t, s_1+s_2)
title('Summed sine waves')
```

Output - Subplot

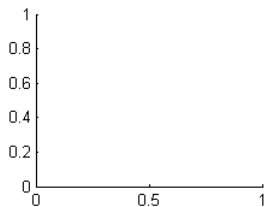
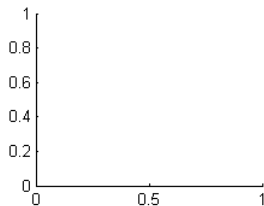
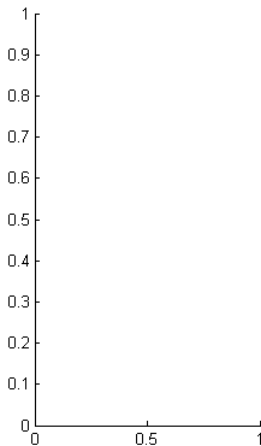


Asymmetric Subplot Example

We can also use subplot to create asymmetric arrangements of plots. For example:

```
figure;  
subplot(2,2,[1 3])  
subplot(2,2,2)  
subplot(2,2,4)
```

Output - Asymmetric Subplot



slice

The other new command we need this week is *slice*. Slice allows use to plot 'slices' of our three-dimensional functions. It is used as follows:

```
figure;  
slice(X, Y, Z, V, sx, sy, sz)
```

slice

The other new command we need this week is *slice*. Slice allows use to plot 'slices' of our three-dimensional functions. It is used as follows:

```
figure;  
slice(X, Y, Z, V, sx, sy, sz)
```

So the following will create a slice of V at $x=2, y=2,$ and $z=-2$.

slice

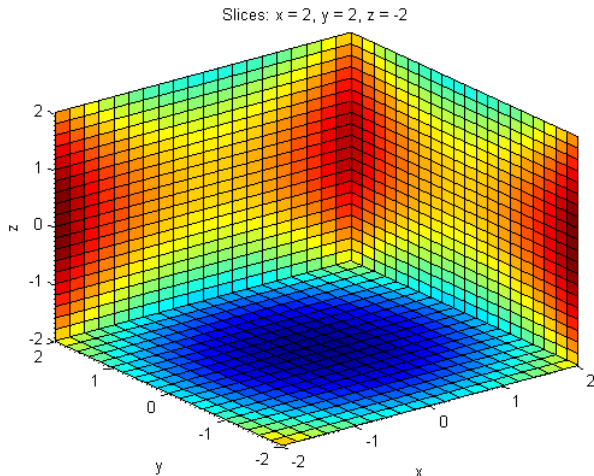
The other new command we need this week is *slice*. Slice allows use to plot 'slices' of our three-dimensional functions. It is used as follows:

```
figure;  
slice(X, Y, Z, V, sx, sy, sz)
```

So the following will create a slice of V at $x=2, y=2,$ and $z=-2$.

```
figure;  
slice(x, y, z, v, 2, 2, -2);
```

Output - Slice Plot



Example 1

In this example we create a 3-D graph of the function $f(x, y) = x^2 + y^2$. First we must initialize our domain:

```
x = -4:0.5:4; y = x;  
[X,Y] = meshgrid(x,y);  
z = X.^2+Y.^2;
```

Example 1

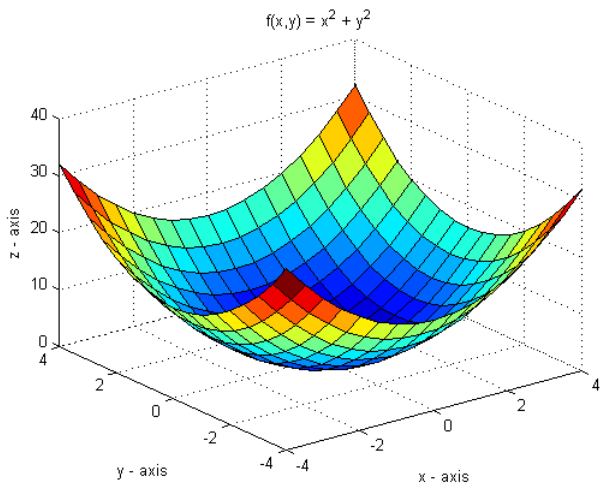
In this example we create a 3-D graph of the function $f(x, y) = x^2 + y^2$. First we must initialize our domain:

```
x = -4:0.5:4; y = x;  
[X,Y] = meshgrid(x,y);  
z = X.^2+Y.^2;
```

Now we use the surf (ie - surface) command to plot the function (and label axes/title):

```
surf(x,y,z);  
title('f(x,y) = x^2 + y^2');  
xlabel('x - axis');  
ylabel('y - axis');  
zlabel('z - axis');
```

Output - Example 1



Graphing Level Curves

Matlab can also handle graphing "level curves," which are 2-dimensional projections of 3-d surfaces. These can be useful in analysis, and visualization.

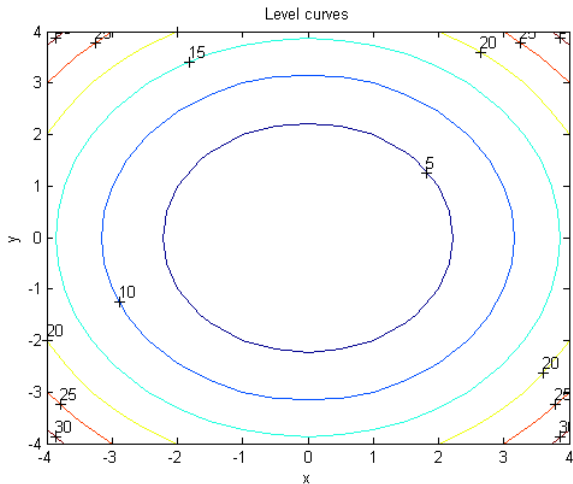
Graphing Level Curves

Matlab can also handle graphing "level curves," which are 2-dimensional projections of 3-d surfaces. These can be useful in analysis, and visualization.

```
x = -4:0.5:4; y = x; [X,Y] = meshgrid(x,y);  
z = X.^2+Y.^2;
```

```
figure;  
C = contour(x,y,z); clabel(C);  
title('Level curves');  
xlabel('x');  
ylabel('y');
```

Output - Level Curves



Level Curves - Customizing the Code

You may have noticed that matlab automatically picks which values of C to graph for the level curves. If you wish to over-ride this and choose your own settings, it is relatively straightforward. You simply pass contour one more parameter which is a vector of constant values to use:

```
figure;  
CC=[1,3,5,7,9,11,13,15];  
C = contour(x,y,z,CC); clabel(C);
```

Level Curves - Customizing the Code

You may have noticed that matlab automatically picks which values of C to graph for the level curves. If you wish to over-ride this and choose your own settings, it is relatively straightforward. You simply pass contour one more parameter which is a vector of constant values to use:

```
figure;  
CC=[1,3,5,7,9,11,13,15];  
C = contour(x,y,z,CC); clabel(C);
```

Or, if you are really daring...

Level Curves - Customizing the Code

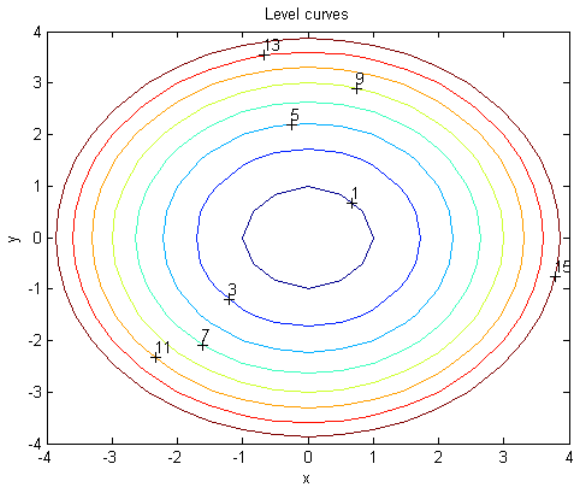
You may have noticed that matlab automatically picks which values of C to graph for the level curves. If you wish to over-ride this and choose your own settings, it is relatively straightforward. You simply pass contour one more parameter which is a vector of constant values to use:

```
figure;  
CC=[1,3,5,7,9,11,13,15];  
C = contour(x,y,z,CC); clabel(C);
```

Or, if you are really daring...

```
figure;  
C = contour(x,y,z,1:2:15); clabel(C);
```

Output - Level Curves



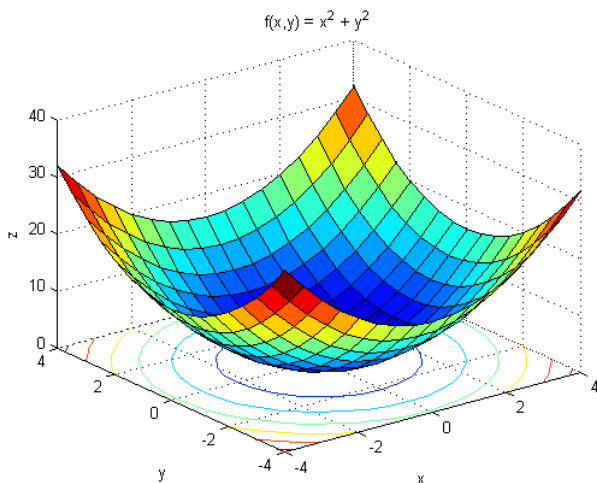
Graphing a Surface with Level Curves

It is sometimes useful to see level curves plotted in the same figure as the 3d surface we are considering. The `surf` command will do just that:

```
x = -4:0.5:4; y = x; [X,Y] = meshgrid(x,y);  
z = X.^2+Y.^2;
```

```
figure;  
surf(x,y,z);  
title('f(x,y) = x^2 + y^2');  
xlabel('x'); ylabel('y'); zlabel('z');
```

Output - Level Curves with a Surface



Level Surfaces

As we move to higher dimensions, level curves become level surfaces. Let us consider a function of three variables, $V = x^2 + y^2 - z^2$. We can define this in matlab using the following:

```
x = -2:0.2:2; y = x; z = x;  
[X,Y,Z] = meshgrid(x,y,z);  
v = X.^2+Y.^2-Z.^2;
```

Level Surfaces

As we move to higher dimensions, level curves become level surfaces. Let us consider a function of three variables, $V = x^2 + y^2 - z^2$. We can define this in matlab using the following:

```
x = -2:0.2:2; y = x; z = x;  
[X,Y,Z] = meshgrid(x,y,z);  
v = X.^2+Y.^2-Z.^2;
```

We would like to consider level surfaces (ie - surfaces defined by $v = c$ where c is just some constant. We can do this using the command *isosurface*:

Level Surfaces

As we move to higher dimensions, level curves become level surfaces. Let us consider a function of three variables, $V = x^2 + y^2 - z^2$. We can define this in matlab using the following:

```
x = -2:0.2:2; y = x; z = x;  
[X,Y,Z] = meshgrid(x,y,z);  
v = X.^2+Y.^2-Z.^2;
```

We would like to consider level surfaces (ie - surfaces defined by $v = c$ where c is just some constant. We can do this using the command *isosurface*:

```
s1 = isosurface(x,y,z,v,-1); % value -1  
s2 = isosurface(x,y,z,v,0); % value 0  
s3 = isosurface(x,y,z,v,1); % value 1
```

Plotting Level Surface Data

The *isosurface* command creates "patch data" which we graph using the function *patch*.

```
figure; %  
p1 = patch(s1);  
p2 = patch(s2);  
p3 = patch(s3);
```

Plotting Level Surface Data

The *isosurface* command creates "patch data" which we graph using the function *patch*.

```
figure; %  
p1 = patch(s1);  
p2 = patch(s2);  
p3 = patch(s3);
```

This function can be customized using the *set* command with various parameters (see below). There are lots more you can check out online as well.

Plotting Level Surface Data

The *isosurface* command creates "patch data" which we graph using the function *patch*.

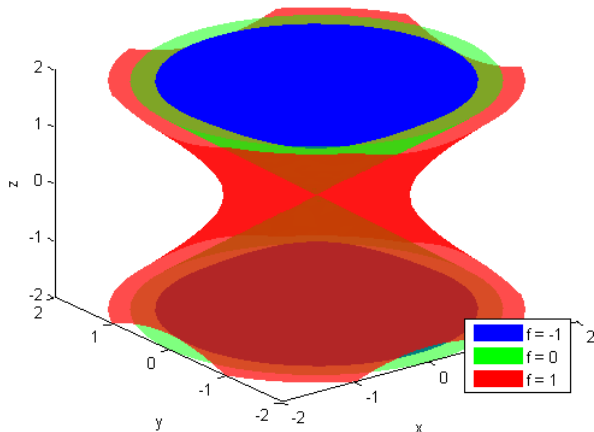
```
figure; %  
p1 = patch(s1);  
p2 = patch(s2);  
p3 = patch(s3);
```

This function can be customized using the *set* command with various parameters (see below). There are lots more you can check out online as well.

```
view(3); % viewpoint specification (3-D view)  
set(p1, 'FaceColor', 'blue', 'EdgeColor', 'none');  
set(p2, 'FaceColor', 'green', 'FaceAlpha', 0.5, 'EdgeColor', 'none');  
set(p3, 'FaceColor', 'red', 'FaceAlpha', 0.7, 'EdgeColor', 'none');
```

Output - Level Surface Output

Level surfaces for $f(x,y,z) = x^2 + y^2 - z^2$



Gradient Function

The gradient of a vector valued function is defined as

$$\nabla F = \frac{\partial F}{\partial x} \hat{i} + \frac{\partial F}{\partial y} \hat{j} + \frac{\partial F}{\partial z} \hat{k}$$

You can think of the gradient as a family of vectors pointing in the direction of greatest increase for F . In matlab, a function called *gradient* computes the numerical equivalent of a gradient. We can even make a nice plot using *quiver*:

Gradient Function

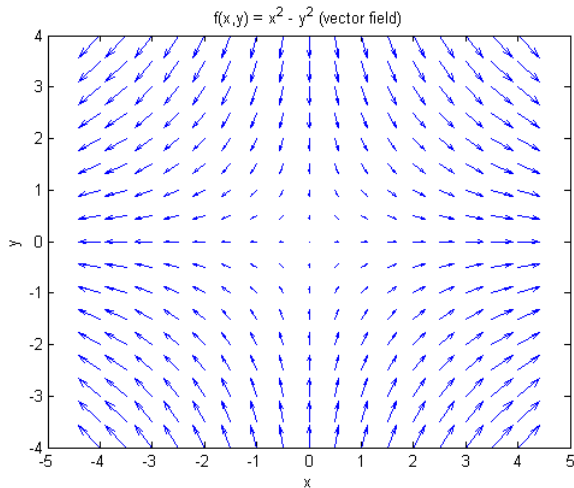
The gradient of a vector valued function is defined as

$$\nabla F = \frac{\partial F}{\partial x} \hat{i} + \frac{\partial F}{\partial y} \hat{j} + \frac{\partial F}{\partial z} \hat{k}$$

You can think of the gradient as a family of vectors pointing in the direction of greatest increase for F . In matlab, a function called *gradient* computes the numerical equivalent of a gradient. We can even make a nice plot using *quiver*:

```
x = -4:0.5:4; y = x;  
[X,Y] = meshgrid(x,y); f = X.^2-Y.^2;  
[Dx,Dy] = gradient(f);  
  
figure;  
quiver(x,y,Dx,Dy);  
title('f(x,y) = x^2 - y^2 (Vector Field)');
```

Output - A Plot of the Gradient



Divergence

Divergence is a *scalar-valued* function which measures the extent to which a point in a vector field acts like a source or sink. It is a local measure of outward flux.

$$\operatorname{div}(\mathbf{F}) = \nabla \cdot \mathbf{F} = \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z}$$

Divergence

Divergence is a *scalar-valued* function which measures the extent to which a point in a vector field acts like a source or sink. It is a local measure of outward flux.

$$\operatorname{div}(\mathbf{F}) = \nabla \cdot \mathbf{F} = \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z}$$

Let's consider a vector field defined as follows:

$$F(x, y) = [xy, x^2]$$

Divergence Code

Matlab has a built-in function to allow easy computation of Divergence. For the function $\mathbf{F}(x, y) = (xy, x^2)$:

```
x = -4:0.5:4; y = x; [X,Y] = meshgrid(x,y);  
div1 = divergence(X,Y,X.*Y,X.^2);
```

Divergence Code

Matlab has a built-in function to allow easy computation of Divergence. For the function $\mathbf{F}(x, y) = (xy, x^2)$:

```
x = -4:0.5:4; y = x; [X,Y] = meshgrid(x,y);  
div1 = divergence(X,Y,X.*Y,X.^2);
```

The command *divergence* accepts four (or six) values which represent the x,y (or z) vectors, and the vector values of the function.

Divergence Code

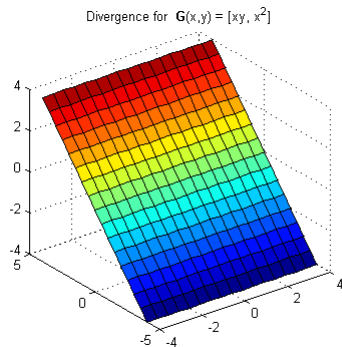
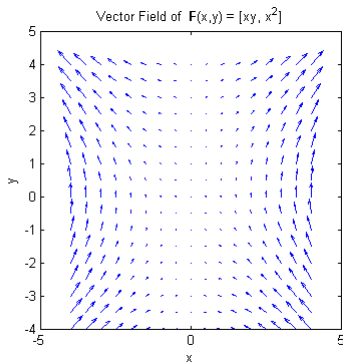
Matlab has a built-in function to allow easy computation of Divergence. For the function $\mathbf{F}(x, y) = (xy, x^2)$:

```
x = -4:0.5:4; y = x; [X,Y] = meshgrid(x,y);  
div1 = divergence(X,Y,X.*Y,X.^2);
```

The command *divergence* accepts four (or six) values which represent the x,y (or z) vectors, and the vector values of the function.

```
figure;  
subplot(1,2,1); quiver(X,Y,X.*Y,X.^2); % Plot the Vector Field  
title('Vector Field of {\bf F}(x,y) = [xy, x^2]');  
  
subplot(1,2,2); surf(x,y,div1); % Plot divergence  
title('Divergence for {\bf G}(x,y) = [xy, x^2]');
```

Output - Divergence of (xy, x^2)



Curl in Matlab

The third important vector property is curl. The curl of a vector field is defined as $\nabla \times F$. Think of the curl as a measure of the angular speed or rotation of the field:

$$\left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z} \right) \mathbf{i} + \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x} \right) \mathbf{j} + \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y} \right) \mathbf{k}$$

In matlab, the command *curl* allows us to compute the curl numerically for a field $\mathbf{F}(x, y, z) = (u, v, w)$:

```
[curlx, curly, curlz, cav] = curl(X, Y, Z, U, V, W)
```

Computing Curl

Let's consider the curl of the vector field defined as:

$$\mathbf{F}(x, y) = (-y, x).$$

```
x = -4:4; y = x; z = x; [X,Y,Z] = meshgrid(x,y,z);  
[cx,cy,cz] = curl(X,Y,Z,-Y,X,zeros(size(X)));
```

Note that if there is no z-values in our field we must supply zeros.

```
figure; %  
quiver3(X,Y,zeros(size(X)),cx,cy,cz,0); hold on; % Plot Curl  
[X,Y] = meshgrid(x,y);  
quiver(X,Y,-Y,X); % plotting vector field F(x,y)
```

Output - Curl of $(-y, x)$

