Advanced Graphics

Last week you learned how to make basic line and scatter plots in MATLAB. This week I'll show you how to make a few more advanced graphics.

Scatter Plots

Scatter plots different that "line" plots because the data points are not connected with a line or assumed to be continous. You can make a normal scatter plot with the regular plot function by specifying a marker type.

```
x = linspace(0,3*pi,200);
y = cos(x) + rand(1,200);
plot(x,y,'.','MarkerSize',10);
```

However, you can also get more advanced by giving the markers a meaningful color or size. This allows the graph to display more relevant information about the data. Here I have let the color be constant but I have scaled the marker size.

```
x = linspace(0,3*pi,200);
y = cos(x) + rand(1,200);
sz = linspace(1,100,200);
scatter(x,y,sz,'b','filled');
```

Here I am scaling both the marker size and the color.

```
c = linspace(1,10,length(x))
scatter(x,y,sz,c,'filled')
```

Finally, the code below scales the color only. I have also included a color bar that tells you what the color of the markers indicate.

```
x = linspace(0,3*pi,200);
y = cos(x) + rand(1,200);
c = linspace(1,10,length(x));
scatter(x,y,[],c,'filled')
colorbar;
```

Plotting 2D and 3D Data

MATLAB can produce graphics that describe multi-dimensional data sets of scalars and vectors. Here I am plotting the voltage around a postive charge (that has been normalized to arbituaray units, i.e. I let any constants equal 1). You might not remember this (or haven't learned it yet) but the slope or gradient of the voltage gives you the electric field. MATLAB can calculate the gradient and make a plot of the electric field vectors or the electric field lines. That is handy!

```
%Meshgrid is a function that is used to define a 2 or 3-dimensional grid of
%points
[X,Y] = meshgrid(-2:0.07:2);
```

```
%Calculate the normalized voltage
V = 1./sqrt(X.^2+Y.^2);

%To make the colors of the plot more reasonable
%I'm removing large values, this function goes to infinity at the origin
V(:,:)=min(V(:,:),1000);

%Draw the contour plot
%Taking the log of the voltage is another trick to
%make the colors of the plot more reasonable. You can
%remove it and see what happens.
figure
subplot(1,3,1)
contourf(X,Y,log(V),10)
axis equal
xlim([-1 1])
ylim([-1 1])

%This tells MATLAB to keep plotting on the same figure
hold on;

%Calculate the gradient
[Ex,Ey] = gradient(V,0.3,0.3);


%The vector electric field is the negative gradient of the voltage
subplot(1,3,2)
quiver(X,Y,-Ex,-Ey,1,'LineWidth',1.)
axis equal
xlim([-1 1])
ylim([-1 1])


%Define some points to start drawing field lines
theta=0:.1:2*pi;

xstart=1.5*cos(theta);
ystart=1.5*sin(theta);

%Use streamline to draw some electric field lines
%I didn't put the negative signs so that the lines would go inward
%instead of outward
subplot(1,3,3)
streamline(X,Y,Ex,Ey,xstart,ystart)
axis equal
xlim([-1 1])
ylim([-1 1])
```

I can also use a surface plot. Here the height of the surface represents the voltage.

```
%Open a new figure an make a surface plot of the voltage
figure
surf(X,Y,V)
view([-63.0 42.1])
```

Animations in MATLAB

MATLAB can be used to make really cool and informative animations. Making movies to describe your science (rather than just graphs) is the future of science. Pretty much all papers are initially viewed online now, so you are stuck with flat paper. Animations are also useful for education.

Below is code from MATLAB's documentation showing a red dot moving accros a sin curve.

```
%Create a list of x and y values to draw the sin curve
x = linspace(0,10,100);
y = sin(x);

%Draw the sign curve
plot(x,y)
hold on

%Plot the red dot and save the plot a variable so you can update the points
%position
p = plot(x(1),y(1),'o','MarkerFaceColor','red');
hold off
axis manual


for k = 2:length(x)
    %Update the points position and update the plot
    p.XData = x(k);
    p.YData = y(k);
    drawnow
end
```

I commented out this code because it only runs in a Script (not in a Live script). The difference between this code and the code above it is that this code produces a movie that you can loop as many times as you want after you produce the data you want to animate.

```
% x = linspace(0,10,100);
% y = sin(x);
% plot(x,y)
% hold on
% p = plot(x(1),y(1),'o','MarkerFaceColor','red');
% hold off
% axis manual
% M(1) = getframe;
```

```
%
%
% for k = 2:length(x)
%     p.XData = x(k);
%     p.YData = y(k);
%     M(k) = getframe;
% end
%
% figure
% movie(M,5)
```

Below I have code for a simple animation of two spheres. One is static and the other is moving accross the space.

```
%Get the coordinates for the surface of a sphere
[X,Y,Z] = sphere;

%Set the radius of each sphere
r1=1;
r2=.5;

for xs=-5:.1:5

    %This clears the figure window, see what happends if you take it out
    clf;

    %Allows both spheres to be drawn on the same image
    hold on;

    %Use surface plots to plot both spheres
    %Notice that the blue sphere (first one) has a changing
    %x position
    surf(r1*(X-xs),r1*Y,r1*Z,'FaceColor',[0 0 1],'EdgeColor','none')
    surf(r2*X,r2*Y,r2*Z,'FaceColor',[1 0 1],'EdgeColor','none')
    hold off;

    %Stuff that controls the axes and view angle of the plot
    axis([-5  5    -5  5    -5  5]);
    axis square;
    view([-20  20]);

    %Draw image once per loop
    drawnow;

end
```