CS 402: Problem Set 2

Directions for Submission

E-mail your answers to me at ajp9@cornell.edu. The subject of your message should be “CIS402 PS2,” and the body of the message should contain your answers. Some mailers can format messages using HTML or RTF. Please turn this feature off and send your message using plain text. If for some reason you cannot send your message as text, you may attach your answers as a text file.

Essential Knowledge–Please give a brief answer (1-2 sentences) for each

1. The function `plotyy` creates two overlapping axes. After you create a plot with this function, how can you change the range of the y-axis on the left?

   Since `plotyy` creates two axes, you must save their handles if you want to change either of them:
   
   ```matlab
   >> [axhand,linhand]=plotyy(...); %captures handles to axes and lines
   >> set(axhand(1),'ylim',...); %axhand=[left axes, right axes]
   ```

   Note: getting the handles from `gca`’s children is also possible, but there’s a catch. If
   
   ```matlab
   >> ax=get(gca,'children')
   ```

   then `ax(1)` will be the axes with the labels on the right—the last axes created. In general, the order of handles in children is the reverse of the order in which the objects were added.

2. You have two data sets `Z1` and `Z2` which you visualize using `pcolor` with the following commands:

   ```matlab
   >> figure(1);pcolor(x,y,Z1);%Assume x and y are appropriate;
   >> figure(2);pcolor(x,y,Z2);
   ```

   You want to ensure that the relationship between color and data value is exactly the same in both figure. Give a few lines of Matlab code to
make the color-mapping is the same in each figure and that the colors span the range of values in both data sets.

The strategy is to get the color limits for both axes, compute the min and max, and set the color limits of both axes to the same value.

```matlab
≫ for j=1:2;figure(j);cl(j,:)=get(gca,'clim');end
≫ cl=[min(cl(:,1)),max(cl(:,2))];
≫ for j=1:2;figure(j);set(gca,'clim',cl);end
```

3. The function **polar** is very strange. Describe how you could change the color of the dotted circles. You don’t need to do this (it’s tedious), just describe a strategy for finding the handles to these objects and changing their color.

Once you have the handles to all of the objects, you need to look for the ones with a linestyle equal to ‘:’. Once you have the handles to these lines, you need to figure out which are the circles and which are the spokes. One approach is to, one-by-one, set their visibility to off.

### Programming

The default shading for **pcolor** is ‘flat’; however, **pcolor** isn’t very good at flat shading. The problem is that flat shading has to pick a single color for each cell, even though each cell is surrounded by four vertices. Matlab’s solution is to use the z-value of the first vertex plotted to determine the color. This means **pcolor** with flat shading discards a whole row and column of data (Figure ??)!

4. Describe why **pcolor** with interpolated shading does not discard any data.

To determine the color of a pixel using interpolated shading, Matlab finds the four vertices of the rectangle surrounding the pixel. The cdata value at the pixel is determined by “bilinear interpolation” from the values at the vertices.

5. Create a new function called **mypcolor.m**. This function should take in vectors x and y and a matrix Z and should return a handle to a surface object. After checking that the sizes of the inputs agree (and returning
intelligent error messages using \texttt{error} if they don’t), your function should create a fake grid described by vectors $x_2$ and $y_2$. You should then create the fake data matrix $Z_2$ and place your data in it. Finally, you should call \texttt{pcolor} with the new variables.

See attached example.