15.11.2018

MAT 116E

HOMEWORK-2

This homework is designed to give you practice with writing functions and visualizing data. This homework will give you more freedom than homework 1 to choose how you implement your functions. As before, the names of hepful functions are provided in **bold** where needed. When you produce your figure, **do not use Matlab's graphic editor. You have to use appropriate graphic options in command form. Otherwise, you will get lower grade.**

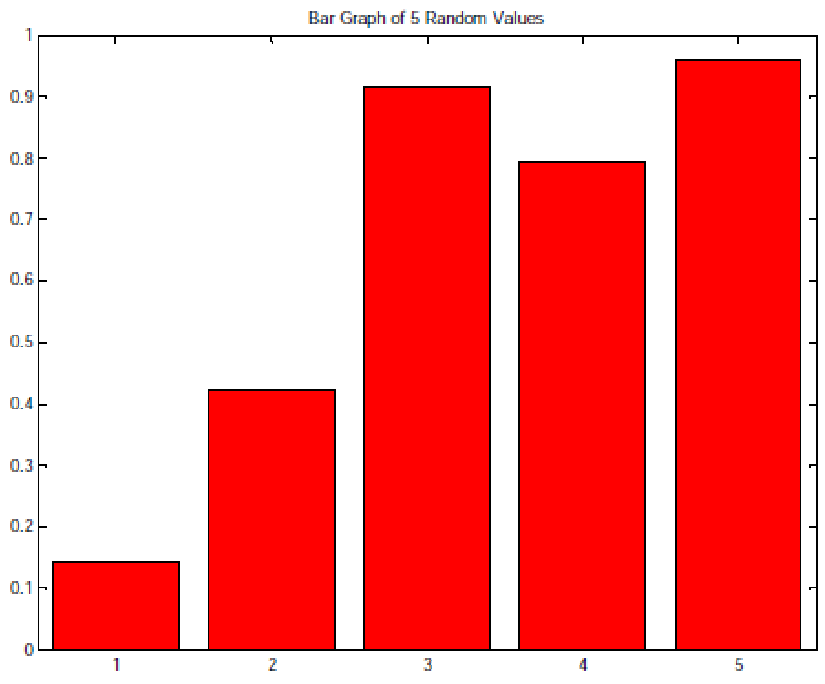
Homework must be submitted on the ninova system after the midterm exam.

What to turn in: Copy the text from your scripts and paste it into a document. If a question asks you to plot or display something to the screen, also include the plot and screen output your code generates. Submit either a *.doc or *.pdf file.

Keep all your code in scripts. If a specific name is not mentioned in the problem statement, you can choose your own script names.

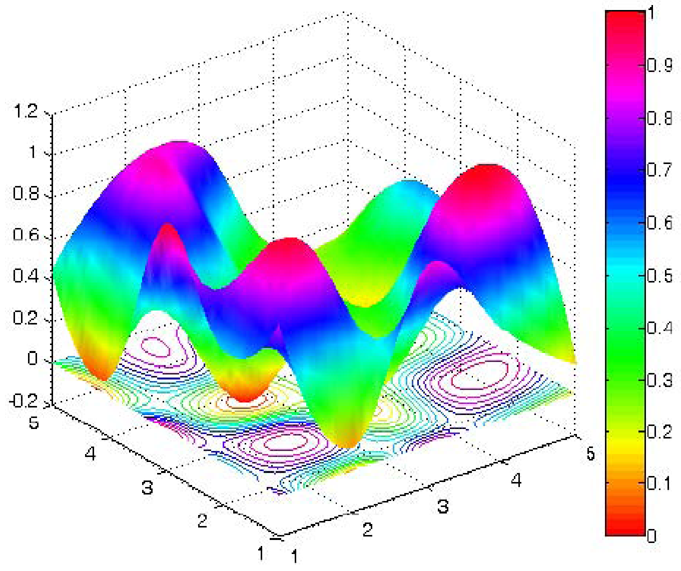
1. Over the past years, the number of students in MAT 232E has been 15, 25 ,55, 115, 144. Class size seems like it's growing exponentially. To verify this, plot these values on a plot with a log y scale and label it (**semilogy, xlabel, ylabel, title**). Use magenta square symbols of marker size 10 and line width 4, and no line connecting them. You may have to change the x limits to see all 5 symbols (**xlim**). If the relationship is exponential, it will look linear on a log plot.

2. Make a vector of 5 random values and plot them on a bar graph using red bars, something like the figure below.



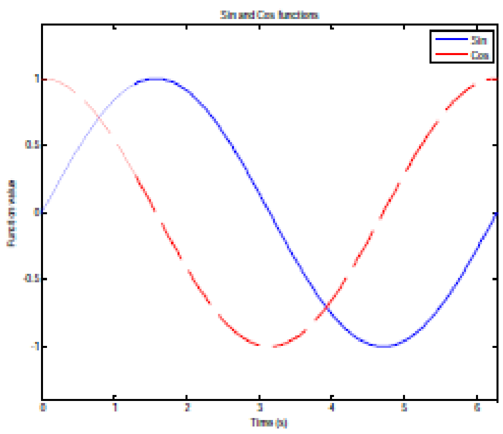
3. Write a script called randomSurface.m to do the following:

a. To make a random surface, make Z0 a 5x5 matrix of random values on the range [0,1] (**rand**).
b. Make an X0 and Y0 using **meshgrid** and the vector 1:5 (use the same vector for both inputs into meshgrid). Now, X0, Y0, and Z0 define 25 points on a surface.
c. We are going to interpolate intermediate values to make the surface seem smooth. Make X1 and Y1 using **meshgrid** and the vector 1:.1:5 (again use the same vector for both inputs into meshgrid).
d. Make Z1 by interpolating X0, Y0, and Z0 at the positions in X1 and Y1 using cubic interpolation (**interp2**, specify cubic as the interpolation method).
e. Plot a surface plot of Z1. Set the colormap to hsv and the shading property to interp (**surf**, **colormap**, **shading**).
f. Hold on to the axes and plot the 15-line contour on the same axes (**contour**).
g. Add a colorbar (**colorbar**).
h. Set the color axis to be from 0 to 1 (**caxis**). The final figure should look something like this (if the figure isn't copy/pasting into your document appropriately, try changing the figure copy options to use a bitmap):



4. In class we covered how to plot a single line in the default blue color on a plot. You may have noticed that subsequent plot commands simply replace the existing line. Here, we'll write a script to plot two lines on the same axes.

a. Open a script and name it `twoLinePlot.m`. Write the following commands in this script.
b. Make a new figure using **figure**
c. We'll plot a sine wave and a cosine wave over one period
   i. Make a time vector t from 0 to 2π with enough samples to get smooth lines
   ii. Plot $\sin(t)$
   iii. Type **hold on** to turn on the 'hold' property of the figure. This tells the figure not to discard lines that are already plotted when plotting new ones. Similarly, you can use **hold off** to turn off the hold property.
   iv. Plot $\cos(t)$ using a red dashed line. To specify line color and style, simply add a third argument to your plot command (see the third paragraph of the **plot** help). This argument is a string specifying the line properties as described in the help file. For example, the string 'k:' specifies a black dotted line.
d. Now, we'll add labels to the plot
   i. Label the x axis using **xlabel**
   ii. Label the y axis using **ylabel**
   iii. Give the figure a title using **title**
   iv. Create a legend to describe the two lines you have plotted by using **legend** and passing to it the two strings 'Sin' and 'Cos'.
e. If you run the script now, you'll see that the x axis goes from 0 to 7 and y goes from -1 to 1. To make this look nicer, we'll manually specify the x and y limits. Use **xlim** to set the x axis to be from 0 to 2π and use **ylim** to set the y axis to be from -1.4 to 1.4.
f. Run the script to verify that everything runs right. You should see something like this:


Sin and Cos functions
Sin
Cos
Function value
Time (s)