CS1112 Fall 2015 Project 1 due Thursday 9/4 at 11pm

You must work either on your own or with one partner. If you work with a partner you must first register as a group in CMS and then submit your work as a group. *Adhere to the Code of Academic Integrity.* For a group, "you" below refers to "your group." You may discuss background issues and general strategies with others and seek help from the course staff, but the work that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is not OK for you to see or hear another student's code and it is certainly not OK to copy code from another person or from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

Objectives

Completing this project will help you learn about MATLAB scripts, assignment statements, if-else statements, and some MATLAB built-in functions. You will also start to explore MATLAB graphics.

Note: Use *scalar* variables only—do not use arrays. Do not use loops.

1 Approximating the cube root

In Lecture 1 you saw that one can approximate the square root of a positive value by constructing "increasingly square" rectangles of the same area. You will now approximate the cube root of a positive value V by constructing increasingly cube-like rectangular solids with volume V.

Start with a rectangular solid of volume V that has two square faces and let s be the side length of the square. The dimensions of the rectangular solid are then $s \times s \times \frac{V}{s^2}$. The side length of the cube with volume V must be a value between the side lengths of the rectangular solid of the same volume. We can use different averages to obtain s for the next, more cube-like, rectangular solid, e.g.,

$$\begin{array}{lcll} s_{\text{new}} & = & \frac{1}{3}(s + s + \frac{V}{s^2}) & \text{option 1: average over three side lengths} \\ s_{\text{new}} & = & \frac{1}{2}(s + \frac{V}{s^2}) & \text{option 2: average over two side lengths} \end{array}$$

Write a script `cubeApprox` that solicits a value V and approximates the cube root of V using the averaging method described above. Begin with side length $s = 1$ and, for each averaging option above, perform five averaging steps on s. Show the results in a table to facilitate comparison, displaying the approximations to four decimal places. To print a "table" simply use `fprintf` with appropriate format sequences so that the values line up neatly. Below is example output (only the first few lines are shown) for $V = .8$.

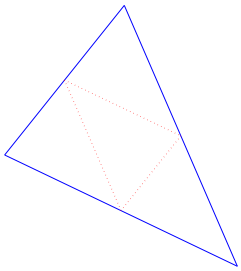
```
Enter a positive value: .8
Estimate the cube root of 0.800000
-----------------------------------------
Step No.   Average over 3   Average over 2
-----------------------------------------
    0          1.0000          1.0000
    1          0.9333          0.9000
    2          0.9283          0.9438
```

At the end of your script, write comments to answer the following questions. Answer in no more than two sentences for each question.

1. Which averaging formula do you prefer? Why?
2. Run your script several times with increasing values for V. How well does this approximation method work with increasing V?

Observation: There probably is quite a bit of copying-n-pasting when you write your program, which is a series of similar-looking assignment and print statements. Soon in the course you will learn how to avoid the tedium by using the programming construct called a "loop."

2 Triangles



Given a triangle, one can partition it into two triangles of equal area by connecting one vertex to the midpoint of the opposite side. A partition into four triangles of equal area can be made by connecting the midpoints of the three sides. You will modify a given script to draw a triangle and possibly partition it based on its area.

Download the file `triPartition.m` and run it. A graphics window showing two points connected by one line will pop up. The message near the top (the title area) says to click in the window. After you click, a black asterisk marks the clicked point and its coordinates are given in the title area.

Read the program to make sure you understand what it does. Don't worry about the early commands to set up the figure window, but here's how the `plot` statement works: `plot(x,y,'ko')` draws a marker at the point (`x`,`y`) with the format "black circle"; `plot([x3 x2],[y3 y2],'b:')` draws a line from the point (`x3`,`y3`) to (`x2`,`y2`) with the format "blue dotted line." Other formats are explained in the program comments. The statement `[x3,y3]=ginput(1)` accepts one mouse click by the user and stores the x- and y-coordinates of the click in the variables `x3` and `y3`, respectively. A statement `title('hello there')` would display the text 'hello there' as the title of a figure. The `sprintf` statement works just like `fprintf` in formating text, but instead of printing directly to the Command Window, `sprintf` allows the text to be saved under a variable name. Then this text (string) variable can be used in other statements, such as the `title` statement as shown in the program.

Now modify the program:

1. Change the fixed locations of the second point (see the comments in the code) to randomly generated coordinates within the interval (1,9) for both x and y. *Hint*: The statement `v = rand` assigns to variable `v` a random number in the range of 0 to 1. So how do you get a random number within a different range? First, the statement `v = rand` gets you a real number in the range of 0 to 1. Next, scale (think multiply) and shift (think add) the value `v` as necessary to get the range you need.

2. After accepting and drawing the user's mouse click, connect the three points (x_1, y_1), (x_2, y_2), and (x_3, y_3) with solid lines to draw a triangle. Use one color of your choice for all three lines of the triangle. Do not draw the markers at the vertices.

3. Compute the area of the triangle. (Forgot the distance and area formulae? See Appendix B in *Insight*, p.413.) Display the calculated area in the title area of the graphic.

4. To partition or not to partition ... Based on the area of the triangle, the program should perform one of these three actions:

   (a) If the area is strictly less than 5 units squared, display the message "Triangle is small!" or something similar in the title area of the graphic and do not partition the triangle.

   (b) If the area is strictly greater than 15, partition into four triangles of equal area by connecting the midpoints of the sides of the original triangle. Use dotted lines in a color of your choice. Note that the midpoint on a line with endpoints (x_1, y_1) and (x_2, y_2) is

   $$\left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2} \right).$$

   (c) Otherwise partition into two triangles of equal area. Draw one solid line in a color of your choice between a vertex and the midpoint of the opposite side. You can choose which vertex to use.

Submit your files `cubeApprox.m` and `triPartition.m` in CMS.