CS 1112 Spring 2020 Project 1 due Tuesday, February 4, at 11:00 PM

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

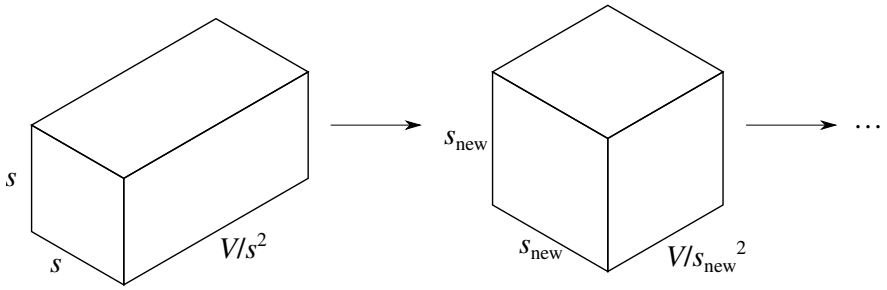
Completing this project will help you learn about MATLAB scripts, branching, and some MATLAB built-in functions. You will also start to explore MATLAB graphics.

Ground Rule

Since this is an introductory computing course, and in order to give you practice on specific programming constructs, some MATLAB functions/features are forbidden in assignments. Only the built-in functions that have been discussed in class or in the assigned reading so far may be used.

1 Approximating cube roots

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 prisms with volume V.



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

$$s_{\text{new}} = \tfrac{1}{3}(s + s + \tfrac{V}{s^2}) \qquad \text{option 1: average over three side lengths}$$

$$s_{\text{new}} = \tfrac{1}{2}(s + \tfrac{V}{s^2}) \qquad \text{option 2: average over two distinct side lengths}$$

Write a script `cubeApprox` that solicits a value V and approximates the cube root of V using the averaging method described above. Begin with square 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 $V = 31$:

```
Enter a positive value: 31
Estimate the cube root of 31.000000
-----------------------------------------
Step No.   Average over 3   Average over 2
-----------------------------------------
    0           1.0000           1.0000
    1          11.0000          16.0000
    2           7.4187           8.0605
      ...
```

At the end of your script, **write comments** to answer the following questions. Be *precise yet concise* (i.e. 1–2 sentences of meaningful reflection, as if you were explaining your results to your manager).

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

Observation: You probably did a lot of copy-pasting when you wrote your program, since it involves a series of similar-looking assignment and print statements. Soon in this course you will learn how to avoid this kind of tedium by using a programming construct called a "loop."

2 Did you click a button?

Download the script `buttonClick.m` and run it. A graphics window will pop up showing two horizontal lines, and a message near the top (the title area) says to click in the window. After you click, the title will change to report the coordinates of your click.

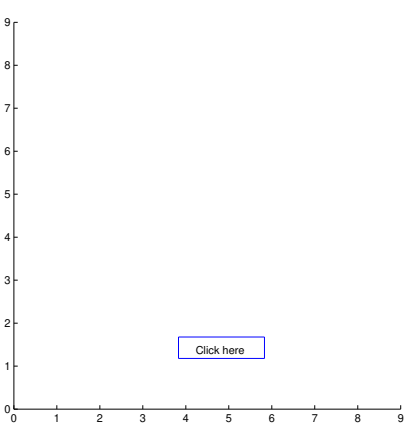
Read the program to make sure you understand what it does. Don't worry about the early commands to set up the figure window, but here's how the `plot()` statement works: `plot([x1 x2],[y1 y2])` draws a line from the point $(x1, y1)$ to $(x2, y2)$. The statement `title('hello there')` displays the text "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()` lets you save the text under a variable name. Then this text (a.k.a. "string") variable can be used in other statements, such as the `title()` statement used near the end of the program.

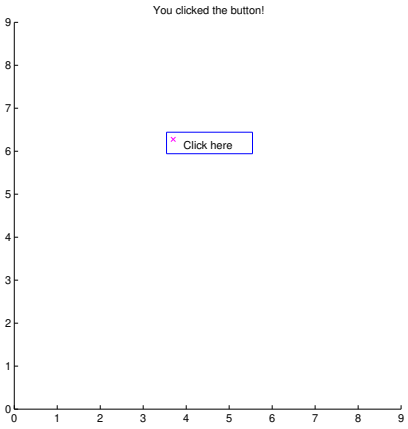
Your job is to **modify this program**:

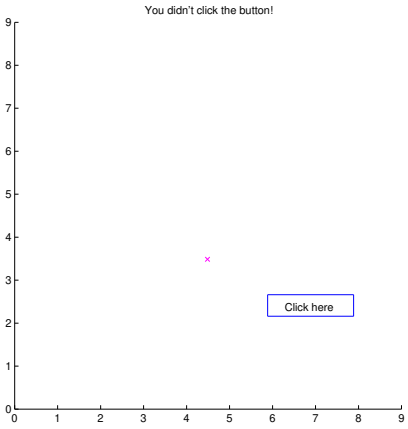
1. Your program should draw a button at a random location anywhere in the window (instead of the two horizontal lines). For this assignment, a "button" is simply a rectangle (four lines) drawn using the `plot()` function. The whole button must lie within the plot area (note that the axis limits and button size are given in the code).
   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 in a different range? Start with a variable in the range of 0 to 1, then scale (think multiply) and shift (think add) its value to get it within the range you need.
2. The button should be labelled with the text "Click here", and the title area should initially be left blank. The built-in function `text()` draws a string in the figure window: `text(a,b,'Click here')` places the the string "Click here" with its left edge centered at the position (a, b). Through experimentation, determine some expressions to calculate `a` and `b` such that the text is approximately centered on the button.
3. Your program should accept a mouse click and mark the location of the user's click. The statement `plot(x,y,'m*')` draws a magenta asterisk at point (x,y). Experiment with other colors and markers! 'g' for green, 'y' for yellow, 'x' for an x, 'o' for a small circle, '+' for, well, you can guess... Use your favorite color and marker!

4. Your program must determine whether the user has clicked on the button, then display one of two messages in the title: "You clicked the button!" or "You didn't click the button!".

Below we show an example of a randomly placed button, an example when the user clicked on the button, and an example when the user clicked off the button:







Additional clarifications:

- We'll be generating random numbers a lot in this course; make sure you (and your partner) understand how to scale and shift ranges!
- Use any color that you like for drawing the button and the user's click; use any marker that you like for the user's click.
- If the user clicks exactly on the boundary of the button, it is up to you whether to consider that "on" (inside) the button. **Document your choice** with a comment.
- The message indicating whether the user clicked on the button should appear in the title area of the figure. One of the messages includes the apostrophe. Given that the apostrophe is also a single quote and the single quote is used to enclose a string, how do you print a message that includes an apostrophe? The solution is to use a single quote followed immediately by another single quote, like this: `title('You didn''t click the button!')`. Note that two single quotes are not the same as a double quote!

3 Quadratic Function

Complete exercise **P1.2.8** in Chapter 1 of *Insight*. Save the file as `myParabola.m`.

Submission

Submit your files on CMS (after registering your group). They should include `cubeApprox.m`, `buttonClick.m`, and `myParabola.m`. Don't accidentally upload the original version of `buttonClick.m` that you downloaded from the course website!