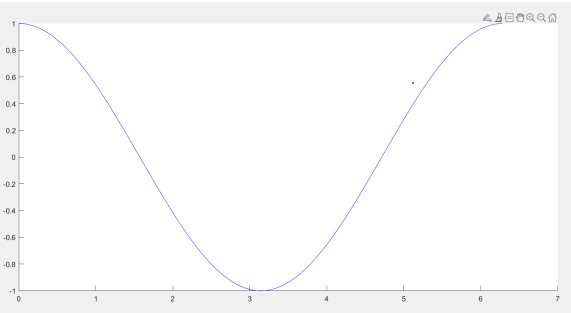
Author: Jahnavi Bonagiri Date : 19 Sept 2022
Bayesian Data Analysis Fall 2022
HW1p: Programming Homework #1
Instructor: Prof. Kevin H. Knuth

A) To create a loop, I first initialized two variables named A and S. Variable A represents the initial starting point of the loop and the S represents the ending iteration. For my for loop condition, I set it from 1 to 10, and added the .10 in a loop for 10 times. After the loop function exits, I used the print statements to display the value of A and the result of subtracting 1 from the final answer.
Results:
Sum : 1.000000e+00
After subtracting 1.0: -1.1102e-16
Based on the results, the loop iteration was correct since my A value was 1.00, however, when subtracting 1.0 from A, I got -1.1 e-16. I was clearly able to see the difference and the error in my opinion was quite large. I believe the reason for these different results when essentially the mathematical equation should be giving the same answer is due to MatLab floating point estimation.

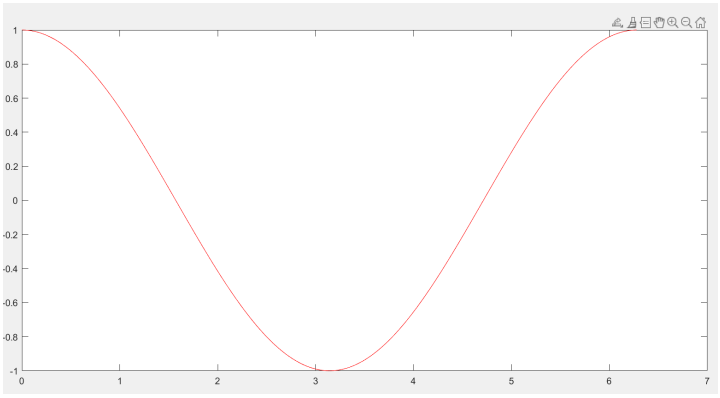
B) Question B was essentially the previous question but instead of using a loop, I used the Matlab Vector functions. I first assigned b (1:10) to equal the value .10, which is what we need to add 10 times. I used the sum function to add the 10 vector values. I subtracted my result by 1.0 again and got the same error as I did with the for loop.
Results:
Sum: 1.0000
After Subtracting 1.0: -1.1102e-16
Elapsed Time of Vector: 0.001126
Elapsed Time of Loop: 0.002015
After displaying the results, I used the functions tic and toc to see the elapsed time it took for the code to finish. The vector code was faster with the elapsed time of .001126 whereas the loop took 0.002015 seconds to complete.

C) To plot the cosine function using the Taylor Series, I first assigned some values to x and used the rem(abs) function to get the absolute value. I used four different cases with if statements so there can be an array of x values to be mapped onto. Once the cases are completed, I compute my function with the use of the Taylor Series. I used 6-7 terms to get an accurate function. Then I used the figure and plot functions to depict my cosine function.

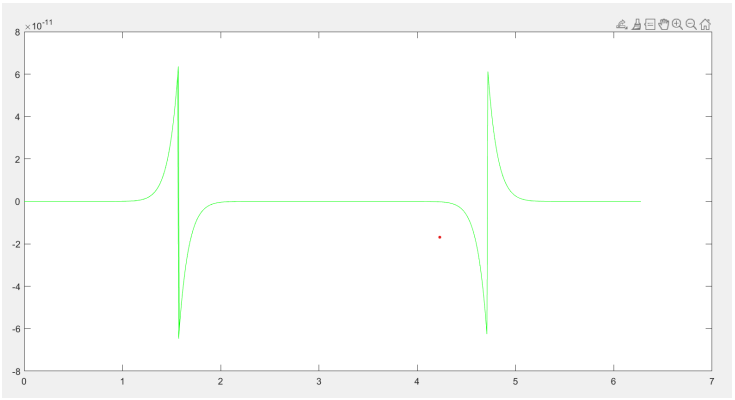


D) For this part, I just plotted the cosine function through MatLab in red. Once plotted, I subtracted the function I computed with MatLab's cosine function in green. The result I got was in a wave format, showing that there is a difference between the two.

MatLab Cosine Function



Difference Plot



E) I computed the probability from a mathematical formula using combinations and also creating a randomized dice simulation with the function randi(). I went about this question by first using the randi function (6, 6, N) creating a matrix of different randomized values. I find the values of 6, sum then and then divide by the N to get the final probability. For the second and third part, it is essentially the same solution however the randi function changes from 6 to 12 or 18 representing the number of dice rolls. I also use the mathematical formula by using combinations. I went about it first by using the probability of no six on a die being (5/6) and raised to the 6th power representing the 6 rolls. Then, by subtracting this result from 1, you get the probability of 1 six in 6 rolls. You repeat these steps for the other parts, adding more combination terms.

The results could differ slightly even though mathematically, they are different proportions due to rounding issues as well as probability being randomized. The results made sense to me. There are many ways to compute the probability. With the 'at least' mentioned in the question, variation occurs as probability is never clearly-cut defined for every roll.

<u>Mathematical Results:</u>

6 Rolls Result: 0.6651

12 Rolls Result: 0.6187

18 Rolls Result: 0.5973