AMS 595
Xiaoyong Sun
112496580

Q1. Base Converter

We want to convert a number from base n to base m. I split the question into two parts. First, convert a number from base n to decimal and the second part will be convert the decimal number to base m. In the end, display the value for the number in base m.

First, I created two prompts to get the values of base n and the number from user. Use a for loop to read through each character in the number and start backwards from the unit's place, tens place, hundreds place and so on, then multiple each character by $n^0, n^1, n^2, \dots$. Sum all the items up to find the decimal value of the number in base n. In this case, the reverse counting the character is because it is easy to read the first number place but difficult to find out the matched n^x value.

Next, I use a prompt to get the value of the base m from user. By definition, I use a while loop to keep dividing "m" to get new quotient instead of the decimal number until the quotient is less than m, then get out of loop. Meanwhile, I collected the remainder from each division and used a string "str" to hold them. After that, 'str' will also hold the value of last quotient which is most significant bit (MSB).

In the end, display the value of string, and it is the number base in m.

Q2. Gaussian Elimination

When I worked on this problem, I first got the input value for the size and each entry of the Matrix A, also asked the user to input the value for column vector b. then create a function called GaussElim. I recall the function to get Matrix A, column vector b and return column vector x as a solution.

In the function GaussElim, I define a vector x1 to hold and return value. Use an if statement to check whether the length of A1 and vector b match or not. If they don't match, then display "No solution for this linear system", else which means if they matched, then combine A1b1 to process the row reduction. I used 3 for loop to convert every two rows 'first entry to be the same, and delete it to be 0, then second ⋯ and so on. Loop through to keep making some non-pivot entries to be 0, at the end I got an upper-triangular matrix. Then use for loop to start from entry in last row and second last column to get the x value. (Last row, last column will be solution for x) and solve until the first x. Return the x1 vector as the solution.

Q3. Approximating Pi

Part a.

First, I receive the input value for the total number of points I want to test for this approximation. I create the function called MontePi to process n of random points and return with three values: approximating Pi, the absolute error, and the relative error.

In the function MontePi, I used a for loop to collect n of points (x, y) with x, y is random number between 0 to 1. Check each point whether inside of the circle with radius 1 or not. If the point is in the circle, then count it into c. Once find out all the points, use it to calculate the value of approximating Pi, the absolute error, and the relative error

Part b.

Got the input value for the total number of points I want to test for the Plots. I used the function MontePi to keep processing the plot problem. Set a “tic” inside of for loop and collect t from “toc” to get the time value, with the number of n to plot the resulting times against n, and use for loop to collect return value of absolute error. Plot the absolute error against the number of random points.