Theodore Ikehara
tikehara@ucsc.edu
16 Jan, 2022

CSE13s Winter 2022
Assignment 2: Numerical Integration
Design Document

About:
In this assignment we are creating a numerical math library containing the functions to compute exponent, sin, cos, log, square root, and numerical integration. The mathematical concepts we will be using to implement these functions will include the Taylor series, Simpson's rule, Newton's method and other concepts that were covered during lecture.

Our Goal:
The goal of this assignment is to make a library that is close to the C math library to further understand and derive the concepts used to create a better understanding of how the math library works and not just take the library works. We create these higher order functions to test and see how any implementation of higher order functions can be implemented with very low level mathematical solutions, in other words this assignment shows us how all functions can be derived just by using the simple plus, minus, multiplication, and division.

Structure:
This program will be structured into functions where each of the functions can be accessed and tested individually. This program will also be started with a getop which allows the user to input different options at the beginning of the program.

The functions that will be Implemented : How these functions will be implemented

- double Exp(double x)
    - This will return the approximate value of e^x
        - In order to implement this function we need to first observe $x^k/k!$however, we also see that $k!$gets large very quickly and can cause our program to run too long or crash. Thus we need

to make the connection and conclude that $\frac{x^k}{k!} = \frac{x^{k-1}}{(k-1)!} \times \frac{x}{k}$ knowing this we can then implement this function with C code with k being the steps and x being the number we are raising to the exponent

- double Sin(double x)
    - This will return the approximate value of $sin(x)$
        - For $sin(x)$ we will be using the Taylor series in order to solve. The Taylor series of $sin(x)$centered about 0 is

          $$sin(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$
- double Cos(double x)
    - This will return the approximate value of $cos(x)$
        - For $cos(x)$ we will be using the Taylor series in order to solve. The Taylor series of $cos(x)$ centered about 0 is

          $$cos(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$$
- double Sqrt(double x)
    - This will return the approximate value of $\sqrt{x}$
        - For $\sqrt{x}$we will be using the Newton's method as we will find that the Taylor series of this function will include square roots thus doing us no good. The Newton's method will look like this

          $$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$
- double Log(doublex)
    - This will return the approximate value of $log(x)$
        - For this log function we will be using the previously created e^x function. And solve for x as $log(e^x) = x$using this principle we can solve for x
- double integrate(double (*f)(double), double a, double b, uint32_t n)
    - This will compute and approximate some function f over the range [a,b] witch will be done using Simpson's ⅓ rule using n partitions.

- The Simpson's ⅓ rule is going to be used in order to solve for this numerical integration. The rule is given by

$$\frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{\frac{n}{2-1}} f(x_{2j}) + 4\sum_{j=1}^{\frac{n}{2}} f(x_{2j-1}) + f(x_n)\right]$$

Pseudo code for mathlib.c

Power (base, raised){
// This if for cos and sin Taylor Series

// This portion accounts for if the power is raised to 0
If raised is = 0
Answer is 1

// This iterates and multiplies the base to itself raised amount of times
For raised
Answer = answer * base
}

Factorial (num){
// for cos and sin Taylor series

If 0! Then return 1

Inverse for loop i--
Answer = answer * i

}

Exp(x)
// use pseudo code in lab man

sin(x){
Split function and solve individually
According to the Taylor series

Loop until error is less than epsilon
}

cos(x){
Same as sin but take out the + 1 in the Taylor series
}

sqrt(x){
Use the pseudo code given in the lab man
}

log(x){
Use the pseudo code given in the lab man
}

integrate(function, low, high, partition){

Get info on function and set low and high values

otherpart = (b - a) / 6

Return answer * (aFunc + bFunc + abFunc)
}