Lucas Lee

luclee@ucsc.edu

Winter 2022

CSE13s Assignment 2 WRITEUP.pdf

Program Analysis

This program implements an integrate function using smaller functions in a mathlib.c file that allows you to calculate the integral of mathematical functions. The program should take in certain command line arguments, which you can see by running the executable file followed by -H as its argument. My integrate function is not working as intended for some of my functions, so gnuplot is unable to properly output acceptable graphs.

Comparing to <Math.h>

```
Exp(0.00000000000000000000) =  +1.000000 (+0.00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Exp(0.10000000000000005551) =  +1.105171 (+0.00000000000000044408920985006261616945266723632812500000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Exp(0.20000000000000011102) =  +1.221403 (+0.00000000000000022204460492503130808472633361816406250000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Exp(0.30000000000000044409) =  +1.349859 (+0.00000000000000022204460492503130808472633361816406250000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Exp(0.40000000000000022204) =  +1.491825 (+0.00000000000000022204460492503130808472633361816406250000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Exp(0.50000000000000000000) =  +1.648721 (+0.00000000000000044408920985006261616945266723632812500000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Exp(0.59999999999999977796) =  +1.822119 (+0.00000000000000022204460492503130808472633361816406250000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Exp(0.69999999999999955591) =  +2.013753 (+0.00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Exp(0.79999999999999933387) =  +2.225541 (+0.00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Exp(0.89999999999999911182) =  +2.459603 (+0.00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Exp(0.99999999999999888978) =  +2.718282 (-0.00000000000000044408920985006261616945266723632812500000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
```

Ex. Comparison of Exp(x) with exp(x)

My Exp function in mathlib.c produces a similar output with acceptable error to the exp function in <mathlib.c>. I implemented my Exp function using the python pseudocode provided in the

assignment document and added base cases to account for Exp(0) and if the given x value is negative.

```
Sin(-3.141592653589793115998) =  -0.000000 (+0.0000000000000000961852408089263566059626089074170723867855046758726378364201
5368968714028596878051757812500000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(-2.641592653589793115998) =  -0.479426 (+0.0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(-2.141592653589793115998) =  -0.841471 (-0.0000000000000001110223024625156540423631668090820312500000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(-1.641592653589793115998) =  -0.997495 (+0.0000000000000001110223024625156540423631668090820312500000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(-1.141592653589793115998) =  -0.909297 (+0.0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(-0.641592653589793115998) =  -0.598472 (-0.0000000000000001110223024625156540423631668090820312500000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(-0.141592653589793115998) =  -0.141120 (+0.0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(0.358407346410206884002) =  +0.350783 (-0.00000000000000005551115123125782702118158340454101562500000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(0.858407346410206884002) =  +0.756802 (-0.00000000000000011102230246251565404236316680908203125000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(1.358407346410206884002) =  +0.977530 (-0.00000000000000011102230246251565404236316680908203125000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(1.858407346410206884002) =  +0.958924 (-0.00000000000000022204460492503130808472633361816406250000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(2.358407346410206884002) =  +0.705540 (+0.00000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sin(2.858407346410206884002) =  +0.279415 (-0.00000000000000005551115123125782702118158340454101562500000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
```

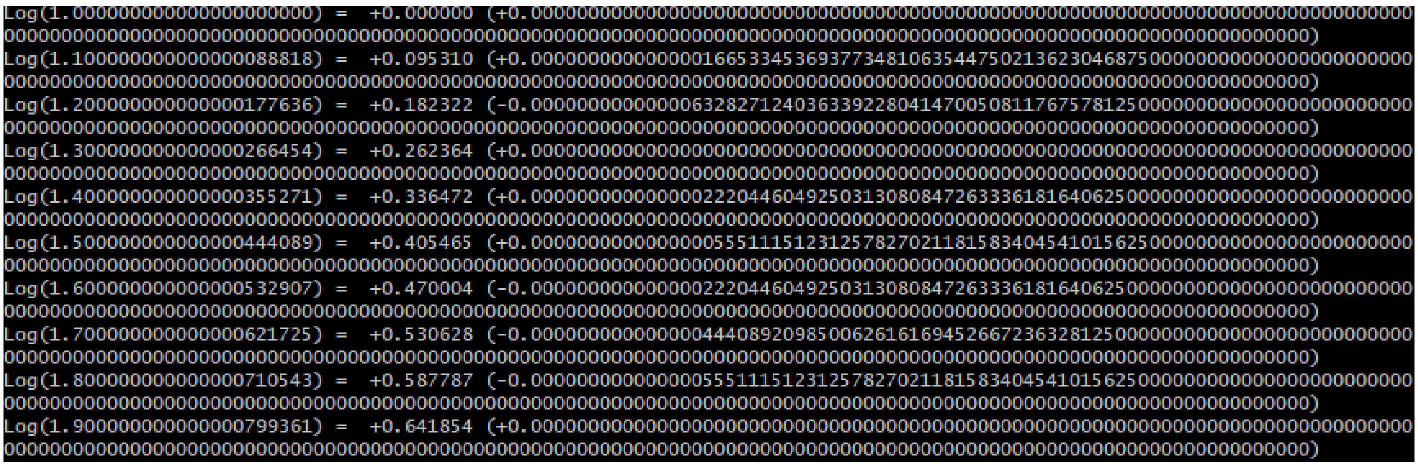
Ex. Comparison of Sin(x) and sin(x)

My Sin(x) function is also within the acceptable error margin of the sin(x) function from the <math.h> library. I did not have a base case for my Sin(x) function, but I did have to implement a scaling option by dividing the given x by 2π until it was a smaller x value with the same Sin(x) value. My Cos(x) function was a mirror of this Sin(x) function but has a different starting point and initial value, so these outputs look similar.

```
Sqrt(0.000000000000000000000) =  +0.000000 (-0.0000000000000071054273576010018587112426757812500000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sqrt(0.100000000000000005551) =  +0.316228 (+0.0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sqrt(0.200000000000000011102) =  +0.447214 (-0.0000000000000000555111512312578270211815834045410156250000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sqrt(0.300000000000000044409) =  +0.547723 (+0.0000000000000001110223024625156540423631668090820312500000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sqrt(0.400000000000000022204) =  +0.632456 (+0.0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sqrt(0.500000000000000000000) =  +0.707107 (+0.0000000000000001110223024625156540423631668090820312500000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sqrt(0.599999999999999977796) =  +0.774597 (+0.0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sqrt(0.699999999999999955591) =  +0.836660 (+0.0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sqrt(0.799999999999999933387) =  +0.894427 (+0.0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sqrt(0.899999999999999911182) =  +0.948683 (+0.0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Sqrt(0.999999999999999888978) =  +1.000000 (-0.0000000000000001110223024625156540423631668090820312500000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
```

Ex. Sqrt(x) compared to sqrt(x)

My Sqrt(x) function also yielded a similar result to the sqrt(x) value from the math library. I followed the assignment doc's pseudo code, and added the scale for this function by dividing x by 4 as many times as possible and multiplying the final sum by 2 for as many times x was divided by 4.



```
Log(1.00000000000000000000) =  +0.000000 (+0.00000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Log(1.10000000000000088818) =  +0.095310 (+0.00000000000000016653345369377348106354475021362304687500000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Log(1.20000000000000177636) =  +0.182322 (-0.00000000000000632827124036339228041470050811767578125000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Log(1.30000000000000266454) =  +0.262364 (+0.00000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Log(1.40000000000000355271) =  +0.336472 (+0.00000000000000022204460492503130808472633361816406250000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Log(1.50000000000000444089) =  +0.405465 (+0.00000000000000005551115123125782702118158340454101562500000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Log(1.60000000000000532907) =  +0.470004 (-0.00000000000000022204460492503130808472633361816406250000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Log(1.70000000000000621725) =  +0.530628 (-0.00000000000000044408920985006261616945266723632812500000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Log(1.80000000000000710543) =  +0.587787 (-0.00000000000000005551115123125782702118158340454101562500000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
Log(1.90000000000000799361) =  +0.641854 (+0.00000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000)
```

Ex. Comparing Log(x) to log(x)

My Log(x) function was close to the log(x) function of the math library as well. I also implemented the scaling option here in the code to account for multiples of e^x . I then added the amount of times this happened to the sum of that function.

My integrate function implemented Simpson's ⅓ rule following a similar format to the given pseudocode given for Simpson's ⅜ rule. I used for loops to run through the summations and I followed the given formula to calculate Simpson's ⅓ rule.

For my integrate function, it only worked for my functions involving e^x, and $\sqrt{x}$. I made it so that the function was able to take in all of the specified command line options, and the -H command to pull up the options menu works, however some of my other functions, when run through my integrate function, did not yield the value I wanted to get. For the functions that work, their

initial values are off, but running them with around 100-1000 partitions will yield a close value to the expected value in the assignment document.

Command line options and other comments

To run the code, the README has the instructions to compile and run, and each command line input specified in the assignment document is able to run. Some of the functions do not have the correct computations(most likely due to some base case problem of some kind?) that I could not resolve. For the functions that have an approximation that approaches the expected outcome needs many partitions.