Jessie Zhu jelzhu@ucsc.edu CSE 13s, Spring 2021 Wednesday, April 14, 2021

Assignment 2: A Small Numerical Library

Design Document

For this assignment we will be creating a small numerical library which will estimate (with high precision) values of a arcsine, arccosine, arctangent, and natural log function. We will also have test file that will print a comparison test between the math.h functions and our implemented functions.

The purpose of this assignment is to understand how to take common math functions and approximate them using Taylor Series (or inverse method, Newton's method) to efficiently and accurately approximate functions.

Design Process:

Overall, I didn't not make large modifications to the design of the lab, but I did add a few thing to increase the accuracy of my functions

- Originally, I had planned to use only a Taylor series for arcsine which would also be used for arccosine and arctangent functions. That part of my design didn't change, but because of the inaccuracy of the Taylor series around |x| = 1, I added something that would check for the value of x for both arcsine and arccosine and if it was too close to 1, I would use a inverse trig identity which would use a different (and thus more accurate) x value for the approximation.

This lab wasn't difficult in terms of the math concepts, but it was challenging learning about how the test file used the mathlib.c file we created separately, as well as how to avoid accidentally truncating a number or infinite sums. For the most part, my errors were all human errors (accidentally dropped a part of a function, missing a variable in important equation).

Helper functions I used:

- Abs (returns absolute value of x)
- Exp (returns value of e^x, from Piazza post @69)
- Sqrt (returns square root of x, from Piazza post @143)

arcSin(x):

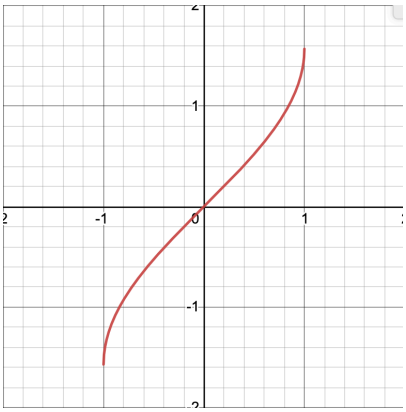
$$\arcsin(x) = x + (\frac{1}{2})\frac{x^3}{3} + (\frac{1 \times 3}{2 \times 4})\frac{x^5}{5} + (\frac{1 \times 3 \times 5}{2 \times 4 \times 6})\frac{x^7}{7} + O(x^9).$$

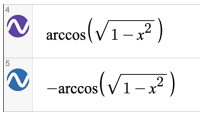
(equation pulled from asgn doc)

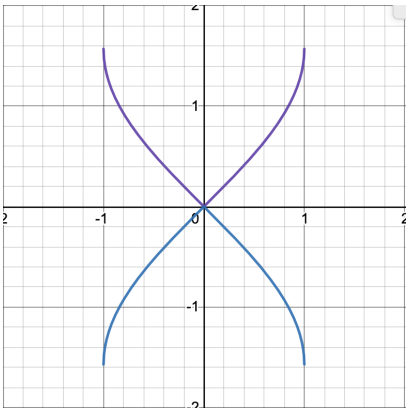
A for loop would be used to handle every new term we are adding to our Taylor series sum. Because of large numbers and truncating, we want to split our numerator and denominator. So for every new term of our sum, we would multiply previous numerator by: (x^2)(exponent minus 2)(exponent minus 2). For the denominator, we would multiply the previous denominator by: (exponent minus 1)(exponent). At the end of the for loop for the an exponent, you would add the numerator/denominator to the sum.

$$\sin^{-1}(x) = \cos^{-1}(\sqrt{1 - x^2}), \ \ 0 \leq x \leq 1$$

(equation pulled from Piazza post @357)







(Graphed on Desmos)

Because arcsine Taylor approximation becomes inaccurate around |x| = 1, we can use the above trig identity and use arccosine to avoid using an x that is close to 1. And as confirmed by above Desmos graphs, when x is close to -1, the equation is slightly different than when x is close to 1.

arcSin(x):

If x is too close to -1 or 1:

calculate arcsin(x) using the appropriate inverse trig identity equation from above and return the approximation

Sum = x

Numerator = x

Denominator = 1

exponent = 3

While the new term is greater than EPSILON:

numerator = (numerator)(k-2)(k-2)(x)(x)

denominator = (denominator)(k-1)(k)

add new term to sum

increment exponent by 2

Return sum

arcCos(x):

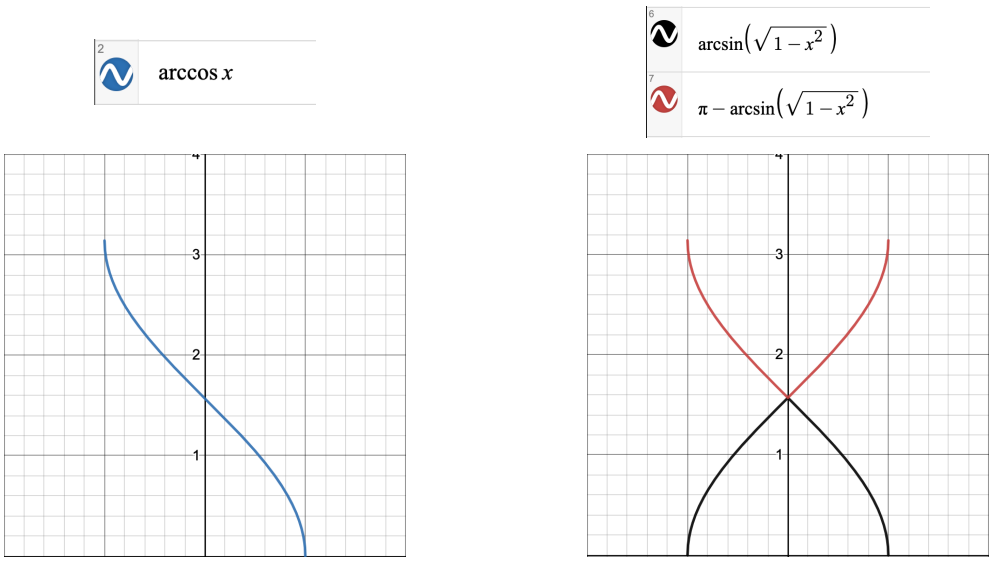
$$\arccos(x) = \frac{\pi}{2} - \arcsin(x).$$

(equation pulled from asgn doc)

We can call the arcSin function we created previously to create our arccosine function using the formula as stated above.

$$\cos^{-1}(x) = \sin^{-1}(\sqrt{1-x^2}), \ 0 \leq x \leq 1$$

(equation pulled from Piazza post @357)



(Graphed on Desmos)

Just like arcsine, Taylor approximation for arccos(x) when x is close to -1 and 1 is very inaccurate. So we can use the above equations on the right for when x is close to -1 and 1, respectively to avoid getting inaccurate estimations.

arcCos(x):

if x is too close to -1 or 1:

calculate arcsin(x) using the appropriate inverse trig identity equation from above and return the approximation

return 2pi minus arc_sin

Jessie Zhu jelzhu@ucsc.edu CSE 13s, Spring 2021 Wednesday, April 14, 2021

arcTan(x):

$$\arctan(x) = \arcsin\left(\frac{x}{\sqrt{x^2+1}}\right) = \arccos\left(\frac{1}{\sqrt{x^2+1}}\right), \quad x > 0.$$

(equation pulled from asgn doc)

We can call the arcSin/arcCos function we created before for our arcTan function using the formulas above.

arcTan(x):

calculate arcsin(x) using the appropriate inverse trig identity equation from above and return the approximation

Log:

$$x_{k+1} = x_k + \frac{y - e^{x_k}}{e^{x_k}}$$

(equation pulled from asgn doc)

For the log function, we will be using Newton's approximation. Using the above equation for Newton's method we can calculate the log approximation. We start the guess at 1, then compare the e^guess to x. While the difference between the two is greater than EPSILON, we will keep calculating the next guess, until e^guess is close enough to x.

(I am following the same structure as the Log function from lecture 9 slides, variable names altered to match previous naming schemes.)

log(x):

guess = 1

exp_guess = Exp(guess)

While distance between x and exp_guess is greater than EPSILON:

    guess = new guess calculated with above equation

    exp_guess = Exp(guess)

Return guess

Test-file:

To manage the user inputs for the test file, we use getopt

There are 5 inputs possible:

- a (test all)
- s (test arcsine)
- c (test arccosine)
- t (test arctangent)
- l (test log)

The input is read, and for every input read, the function will mark which tests need to be printed. Then at the end, all the required tests will be printed and no repeats. If none of the tests are marked, then the user didn't enter an input or didn't enter a good input, which will then display the input options and return an error.

Main:

Opt = 0

Create 4 boolean variables for each function (keep track of what to print)

While object read is of legitimate input type:

case a, make all 4 booleans true

case s, make arcsine boolean true

case c, make arccosine boolean

case t, make arctangent boolean true

case l, make log boolean true

If all booleans are false, print all options, return error

If s is true, print arcsine tests

If c is true, print arccosine tests

If t is true, print arctangent tests

If l is true, print log tests

Return success