Mattiwos Belachew
mbelache@ucsc.edu
4/18/2021

CSE 13s Spring 2021
Assignment 2:
A Small Numerical Library
Design Document

Description:

I will make a small numerical library which contains arcSin, arcCos, arcTan, and Log using addition, subtraction, multiplication, and division which a computer typically has unless its a special-purpose processor. These functions are used in nearly every program that we write, so we ought to understand how they are created.

Files:

- Mathlib.h
  - Contains the function prototypes for the math functions I am required to implement.
- WRITEUP.pdf
  - It contains a discussion of the results for my tests. Specifically, analysis of the differences in the output of my implementations vs. math.h library.
- Makefile
  - Builds, cleans and debugs the program, lrc.c.
- DESIGN.pdf
  - The design document describes purpose, covers the layout, clear description of program parts, and pseudocode.
- mathlib.[c]
  - This file contains my math function implementations, as prototyped in mathlib.h.
- README.md
  - Instruction page for building and running the program.
- mathlib-test.[c]
  - This file contains the main() program and acts as a test harness for my math library.

TOP LEVEL:
The top level design of the code is given by the following pseudocode:

```
Main:
        Opt = 0
Options = “actl”
        while  ((opt = getopt(argc, argv, OPTIONS)) != -1) { //program argument parser.
                Switch (opt) {
                Case ‘a’: //run all
                        Statement
                        Break;
                Case ‘s’: //arcsin
                        For (i = -1.0; i >1; i +=.1){
                        print(arcsin(i))
                        }
                        Break;
                Case ‘c’://arccos
                        For (i = -1.0; i >1; i +=.1){
                        print(arccos(i))
                        }
                        Break;
                Case ‘t’://arctan
                        For (i = 1; i <10; i +=.1){
                        print(arctan(i))
                        }
                        Break;
                Case ‘l’: //log
                        For (i = 1; i <10; i +=.1){
                        print(log(i))
                        }
                        Break;
                }
                Return 0
        }
```

arcSin():

```
        Takes in x as a parameter;
        If (abs(1-x) < EPSILON){ //Trig Identity
                Return arcCos(Sqrt(1-(x*x))
```

```
}
Else if (abs(-1-x) < EPSILON){ //Trig Identity
Return -arcCos(Sqrt(1-(x*x))
}
Sum =x
numerator = 1
Denominator = 2
term = 1 //n/d
Powx = x
While (abs(term*(powx/numerator)) > EPSILON/10)
        term *= numerator/ Denominator
        numerator+=2
        Denominator+=2
        Powx*= x*x
        sum+= term * (powx/numerator)
Return sum
```

- In order to find arcSin(x) I first found the Taylor series of arcsin and wrote a program which corresponds to the mathematical equation in order to find the approx.

arcCos():

```
Takes in value x as parameter
i  = value of arcSin(x) function
Return ((pi/2) - i )
```

- In order to find arcCos(x) I used the fact that the difference between arccos and arcsin is pi/2 so I took that into account and wrote the program. This allowed me not to waste my time writing a Taylor series program for arcCos but just used arcSin's instead.

arcTan():

```
Take in value x as a parameter
Denominator = sqrt(pow(x,2) +1)
Return arcSin(x/denominator); //when x >0
```

- I used trig identity in order to come up with arcTan by using my arcSin function.

Log():

```
Take in value x as parameter
y_k = 1.0 //guess
f = Exp(y_k);
While (abs(x-f) > EPSILON) {
        Y_k += (x-f)/f
```

```
        f = Exp(y_k)
    }
    Return y_k
```

- I used newton's method to come up with Ln(x) following the Eric Hernandez document as reference in order to better understand how to use newton's method. It first begins with a guess, tests it by comparing it to the goal x and adjusts the guess in order to get a closer and closer approximation.

Exp():

```
    Takes in a value x as parameter
    //e^x // By Prof. Long. found on piazza and slides
    Term, sum = 1
    For (k = 1; abs(term) > EPSILON; k++){
        Term *= x/k
        Sum += term
    }
    Return sum
```

- This uses Taylor series in order to come up with the approximation of e^x.

Design Process:

1. One of the biggest problems I encountered was finding the value of -1 and 1 for arcSin because you lose accuracy as you get closer to 1 which causes the approx. to differ a lot. In order to fix this I used trigonometric identity which solved my problem and significantly reduced the difference between my library and math.h library.

What I learned:

1. I learned how to implement mathematical equations and how math libraries work behind the scenes only using the available tools such as addition and subtraction, division and multiplication in order to solve for arcCos, arcSin, Log, and arcTan.
2. I learned that computers aren't that accurate and you're never able to get perfect answers but can only find approximate answers.
3. I also learned that for some reason I cant define a abs function using #define because it messes up with the data.

Resources Used:

- Asgn2.pdf
- asgn5Design by William Santosa from Piazza
- Example_design.pdf found in CSE 13s Discord.
- E_H_Newton's Doc post by Eroc Hernandez