Merilyn Kuo
CSE 13S
Spring 2021

Assignment 2: A Small Numerical Library
Design Document

Purpose:

The purpose of this assignment is to learn how libraries are created and understand how the functions that libraries have are not made of magic. With this assignment I learn how to create functions to calculate arcsin, arccos, arctan, and ln. I practice using loops and if statements. I also learn how to use getopt() and how to parse through the arguments with a switch case statement.

Program Files:

`mathlib.c`: This file will contain the function implementation for arcSin, arcCos, arcTan, and Log as well as any supporting functions for square root and absolute value. The functions will be implemented with Newton's Method.

`mathlib-test.c`: This file is the test harness. The program will allow us to use command line options to print out results from testing the math functions. The program should loop through a specified range for x and print out the outputs of arcsin, arccos, arctan, and log using the functions created in `mathlib.c` and that of `math.h` library. It should also print out the computed difference between them.

Program Setup:

`mathlib.c`:
Include:
- stdio.h
- mathlib.h

Define:
- epsilon: 1e-10

`mathlib-test.c`:
Include:
- stdio.h
- math.h
- unistd.h
- mathlib.h

Define:
- options: asctl

Program Implementation:

All of the functions can be computed with Newton's Method. The general iterative formula for Newton's Method is:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

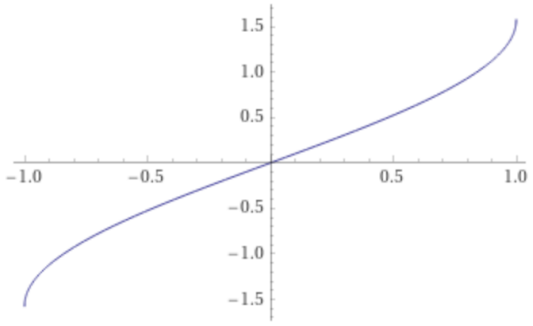
Note that $f(x_k)$must be a differentiable function and $x_k$is the initial approximation for $f(x) = 0$. However for the program, I want to find some y that is the result of $f(x)$ so $f(x) = y$. Therefore the iterative formula I will use is:

$$y_{k+1} = y_k - \frac{f(y_k)}{f'(y_k)}$$

Now, to write my arcsin function, I can use $y = arcsin(x)$. Sine we can't use arcsin in our algorithm, we need to manipulate the formula to be $x = sin(y)$. Since we are finding an approximation for when the function is equal to 0, we solve $x = sin(y)$ for 0 and get $f(y) = sin(y) - x$. Following this the iterative algorithm for the arcSin function is:

$$y_{k+1} = y_k - \frac{sin(y_k) - x}{cos(y_k)}$$

To determine the initial guess, y_k, we can plot the arcsin function with the domain [-1,1].



The actual value of arcsin(x) is always close to x itself so I will use x as the initial guess. I will use a while loop to continue approximating until the difference between approximations is small enough.

With all this, the pseudocode for the arcSin function is as follows:

```
arcSin (x):
    yk = 0, yk_1 = x
```

```
    while(|yk_1 - yk| > epsilon):
        yk = yk_1
        yk_1 = yk - (sin(yk) - x)/cos(yk)
    return yk_1
```

For the arccos function, I can use the identity, $arccos(x) = \frac{\pi}{2} - arcsin(x)$. Using such, the pseudocode for arcCos is as follows:

```
arcCos (x):
    return (pi/2) - arcSin(x)
```

Arctan can also be computed with the identity, $arctan(x) = arcsin(\frac{x}{\sqrt{x^2+1}})$. So, the pseudocode for arcTan is:

```
arcTan (x):
    return arcSin(x/√(x*x) + 1))
```

The implementation of the Log function will be very similar to the implementation of the arcSin function. I used the same Newton's Method formula:

$$y_{k+1} = y_k - \frac{f(y_k)}{f'(y_k)}$$

The equation we are using is $y = log_e(x)$ and converted to exponent form, $e^y = x$. Solving for zero we get $e^y - x$ which will be $f(y_k)$. Therefore our iterative algorithm is:

$$y_{k+1} = y_k - \frac{x - e^{y_k}}{e^{y_k}}$$

The initial guess will be 1.0. The following is the pseudocode for Log:

```
Log (x):
    yk = 0, yk_1 = x
    while(|yk_1 - yk| > epsilon):
        yk = yk_1
        yk_1 = yk + (x - Exp(yk))/Exp(yk)
    return yk_1
```

Note that some of these functions need to use square root, e^x, and/or absolute value. I will be using the Sqrt() function and Exp() function provided on Piazza. For absolute value, I will be

writing my own function with this pseudocode:

```
AbsVal(x):
    if x < 0:
        x = x * -1
    return x
```

Test Harness Pseudocode:

Since the program will need a table header a few times, I will use a function for creating the table header.

```
printTableHeader(function name):
    print(“  x        function name        Library        Difference”)
    print(“-----      -------------        -------        ----------”)
```

The code for the main function is based on the example program for parsing options with getopt() in the assignment PDF and the code Sahiti went through in her section. The formatting of the print statement for printing the outputs was also given in the assignment PDF.

For parsing the arguments, I can use a switch case. Based on the variables after looping through all the arguments, I use if statements to check which outputs to print.

```
main (argc, **argv):
    opt = 0
    while opt = getopt(argc, argv, options) != -1:
        switch (opt):
        case a:
            arcsin = 1
            arccos = 1
            arctan = 1
            log = 1
        case s:
            arcsin = 1
        case c:
            arccos = 1
        case t:
            arctan = 1
        case l:
            log = 1
        default:
            print error message
```

```
        return 1
if arcsin == 1:
    printTableHeader(arcSin)
    for i in range -1 to 1 with step of 0.1:
        printf(" %7.4lf %16.8lf % 16.8lf % 16.10lf\n",
        i, arcSin(i), asin(i), arcSin(i) - asin(i))
if arccos == 1:
    printTableHeader(arcCos)
    for i in range -1 to 1 with step of 0.1:
        printf(" %7.4lf %16.8lf % 16.8lf % 16.10lf\n",
        i, arcCos(i), acos(i), arcCos(i) - acos(i))
if arctan == 1:
    printTableHeader(arcTan)
    for i in range 1 to 10 with step of 0.1:
        printf(" %7.4lf %16.8lf % 16.8lf % 16.10lf\n",
        i, arcTan(i), atan(i), arcTan(i) - atan(i))
if log == 1:
    printTableHeader(Log)
    for i in range 1 to 10 with step of 0.1:
        printf(" %7.4lf %16.8lf % 16.8lf % 16.10lf\n",
        i, Log(i), log(i), Log(i) - log(i))
```

Makefile:

The makefile should have the following commands:

- `all`: builds `mathlib-test` program
- `clean`: remove files the compiler made
- `clang-format`: formats all .h and .c files

Process and Changes to Design:

1. I created a variable for arguments in the test harness file. The variable is zero if the user entered no arguments and 1 if the user enters some argument. Then there is an if statement after the while loop to check if the variable is zero. If so, it will print the error message.

Credits and Resources Used:

Newton's Method Video: https://youtu.be/-5e2cULI3H8
exp.c and sqrt.c posted by Professor Long on Piazza.
Sahiti and her section
The C Programming Language 2nd Edition