Zack Traczyk
ztraczyk@ucsc.edu
Professor Long
03/30/21

CSE13S Spring 2021
Assignment 2 - A Small Numerical Library

DESIGN DOCUMENT

OBJECTIVE:

Create a numerical library and a test harness. Functions for arcsin, arccos, arctan, and log functions need to be implemented. For the trigonometric functions implement them using Taylor series approximations or the inverse method. For the log functions use Newton's method. Then in the test harness compare your implemented functions to the corresponding implementations in the standard library `<math.h>` and output the results into a table.

GIVEN:

- EPSILON = $1x10^{10}$
- Allowed to use `sin()` and `cos()` from `<math.h>`
  - Use `M_PI` for π
  - Use `M_PI_2` for π / 2
- Helper functions written by prof Long:
  - Float absolute value
  - Square root
  - E^x

ARCSIN AND ARCCOS AND ARCTAN:

The program calculates the value of arcsin, then uses various trigonometric identities in order to calculate arccos and arctan. To calculate arcsin the following Mclaurin series was used:

$$arcsin(x) = \sum_{k=0}^{\infty} \frac{(2k)!\, x^{2k+1}}{2^{2k}(k!)^2 2k+1}, \ |x| \leq 1$$

However, computing on a term by term basis quickly becomes infeasible as calculating factorials and exponents becomes expensive fast. Instead, it is more efficient to compute each new term using the last term, thereby removing redundant computations. The pattern can be seen by looking at the first four non-zero terms of the series:

$$arcsin(x) = x + (\frac{1}{2})\frac{x^3}{3} + (\frac{1\times3}{2\times4})\frac{x^5}{5} + (\frac{1\times3\times5}{2\times4\times6})\frac{x^7}{7}$$

Therefore, if k is the index of non-zero terms (starting with x as k = 0), then the next term can be calculated by doing three things:

1. Multiply the last term by X^2
2. Multiply the last non-zero term by $\frac{2k-1}{2k}$
3. Multiply by 2k and divide by 2k + 1, or store the value and increment by 2 (how it is implemented)

As x gets further from the center, the approximation becomes less accurate with a finite number of terms. To account for this, the value of x is translated and computed for arccos when it is closer to 1 since arcsin(x) = arccos($\sqrt{1-x^2}$). The accuracy of the approximations is extremely apparent when looking at the maclaurin series with two terms.

For example:

True Value

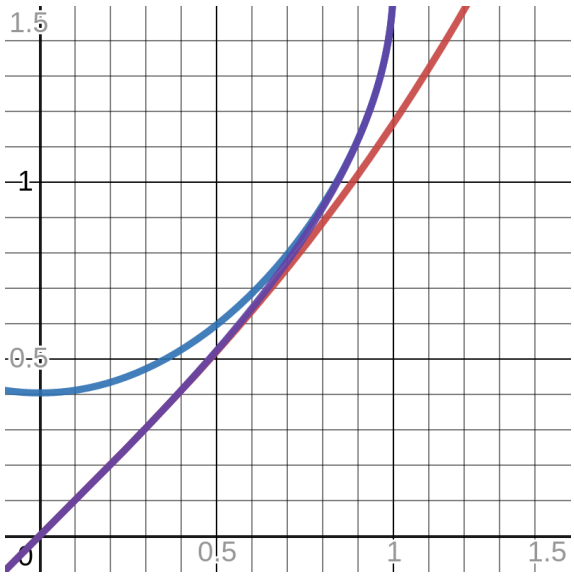
$$arcsin(x)$$

Arcsin Series

$$x + \frac{x^3}{6}$$

Arccos Series translated

$$\frac{\pi}{2} - \left(\left(\sqrt{1-x^2}\right) + \frac{\left(\sqrt{1-x^2}\right)^3}{6}\right)$$



Looking at the graph, the arcsin series is more accurate for calculating arcsin from 0 to ~0.7, while arccos with x translated is more accurate from ~0.7 on. Therefore for the `arcSin` function, the arcsin mclaurien series is used when the absolute value of x is less or equal to 0.7 and the translated arccos maclaurin series is used for the rest of the values.

With arcsin implemented `arcCos` and `arcTan` are trivial since arccos(x) = $\frac{\pi}{2}$ - arcsin(x), and arctan(x) = arcsin(x / $\sqrt{x^2+1}$).

LOG:

$Log_e(x)$ is computed using newton's method:

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

Remember that $Log_e(x) = y$ is equivalent to $e^y = x$ which is easier to deal with. This makes $f(x) = e^y - x$ and thus $f'(x) = e^y$. Knowing these values, makes newton's method easy to compute:

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

This can be simplified further:

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

PSEUDOCODE:

Knowing the mathematical basis to the code the pseudocode is pretty straight forward. All datatypes are `double` unless specified otherwise.

mathlib.c

```
FUNCTION arcSin TAKES x:
    output = 0
    x_squared = x^2
    int use_arccos = fabs(x) > 0.7;

    IF use_arcos:
        x = Sqrt(1 - x^2);
        output -= π/2 // use math standard library for pi/2

    numerator = x
    denominator_factorial = 1;
    term = x
    output += x
```

```
    FOR ITERATE int k=0 BY 2 UNTIL fabs(term) < EPSILON:
        numerator *= (k - 1) * x_squared
        denominator_factorial *= k
        term = numerator / (denominator_factorial * (k + 1))
        output += term

    IF correct:
        RETURN -output
    ELSE:
        RETURN output

FUNCTION arcCos TAKES x:
    RETURN π/2 - arcSin(x)

FUNCTION arcTan TAKES x:
    RETURN π/2 - arcSin(x / sqrt(x^2 + 1))

FUNCTION Log TAKES x:
    guess = x

    DO:
        previous = guess
        guess = previous - 1 + (x / Exp(previous))
    WHILE fabs(guess - previous) > EPSILON

    RETURN π/2 - arcSin(x / sqrt(x^2 + 1))
```

mathlib-test.c

```
FUNCTION tests TAKES function operation, function library, string name
                     start, end:
    PRINT table header
    FOR start TO end BY interval:
        PRINT x, this library, stand library, difference

MAIN:
    IF arguments < c:
        PRINT ENTER AN OPTION
```

```
FOR EVERY ARGUMENT:
    IF 'a': DO ALL
    IF 's': ARCSIN
    IF 'c': ARCCOS
    IF 't': ARCTAN
    IF 'l': LOG
```

RESOURCES:

- Passing function pointers as parameters to functions
    - https://codeforwin.org/2017/12/pass-function-pointer-as-parameter-another-function-c.html#:~:text=In%20C%20programming%20you%20can,another%20function%20using%20function%20pointers.