Fabrice Kurmann
fkurmann@ucsc.edu
12 April, 2021

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

**Assignment:** Create a numerical library (arcsin, arccos, arctan, log). Use taylor series approximation or the inverse method for the first three and Newton's method (for e^x inversed) for log.
Create a test harness in which this library will be compared to math.h

**General Guidelines:**
Test arcsin, arccos for [-1,1), step = 0.1
Test arctan and log for [1, 10), step = 0.1

Separate functions for each implementation
- arcSin()
- arcCos()
- arcTan()
- Log()

Halt computation at an epsilon value of 10^-10

**Library File Math, functions, and pseudocode:**

Log using Newton's method:

Newton's method in general: $X_{K+1} = X_K - \frac{f(x_K)}{f'(x_K)}$

$$X_{K+1} = X_K + \frac{Y - e^{x_K}}{e^{x_K}}$$ , Y = initial input, point to solve for

Ex) input = 2

should approach 0.693147

$X_0 = 1$

$x_1 = \left(1 + \frac{2 - e^1}{e^1}\right) = 0.735$

$x_2 = \left(0.735 + \left(\frac{2 - e^{0.735}}{e^{0.735}}\right)\right) = 0.69477$

$x_3 = \left(0.69477 + \left(\frac{2 - e^{0.6948}}{e^{0.6948}}\right)\right) = 0.69312$

Convergence !

Taylor approximation pseudocode for Log:

log (double Y):
epsilon = $10^{-10}$ ———— accuracy threshold
———— step size of a particular iteration
└ (once smaller than ε, we have precision)
value = 1 ———— running value of function
└ initially 1 (on iteration 0)
for (int K = 1 step_size > epsilon, K += 1): ———— K = step counter

step_size = $(Y - e^{value}) / e^{value}$ * — Exp(function needed)

value = value + step_size

return value

ArcSin (and arcCos) using Taylor series:

Taylor series for arcsin centered at 0:

$$\sum_{k=0}^{\infty} \frac{(2k)!}{2^{2k}(k!)^2} \cdot \frac{x^{2k+1}}{2k+1}, \quad x \geq 1$$

$$= x + \left(\frac{1}{2}\right)\frac{x^3}{3} + \left(\frac{3}{8}\right)\frac{x^5}{5} + \left(\frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6}\right)\frac{x^7}{7}$$

k = 0 k = 1 k = 2 k = 3

Ex)

input = 0.75

should approach 0.8481 convergence

k = 0 : 0.75
k = 1 : 0.82
k = 2 : 0.838
k = 3 : 0.849

For arccos centered at 0:

$$\frac{\pi}{2} - \arcsin x = \text{value}$$

Ex) input = 0.4, should approach 1.1593

k = 0 : $\frac{\pi}{2} - 0.4 = 1.17$
k = 1 : 1.16
k = 2 : 1.1594

convergence

Note, values near 1 require additional trig properties in order for error not to grow.

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

Credit: CSE13 Piazza

try $\sin^{-1}(x) = \frac{\pi}{2} - \sin^{-1}\left(\sqrt{1-x^2}\right)$ when x = 1, this makes 1 not show up in the direct sin computation.

Thus :

if abs(x) > 0.8, use above sin implementation (requires storing sign for neg.)

Taylor approximation **pseudocode** for arcSin and arcCos:

```
arcsin(double x):
    epsilon = 10^-10          ——— accuracy threshold
    step_size = x             ——— step size of a particular iteration
                                  └ (once smaller than ε, we have precision)
    value = x                 ——— running value of function
                                  └ to be returned
                    ┌ 0! = 1
    for (int K = 1, step_size > epsilon, K += 1):   ——— K = step counter
        step_size = step_size ((2K-1)(2K-1) / ((2K)(2K+1)) x^2)
        value = value + step_size
    return value
```

```
arccos(double x):
    epsilon = 10^-10
    step_size = x
    value = π/2 - x           ——— arccos starting point
                                  differs from arcsin.
    for (int K = 1, step_size > epsilon, K += 1):
        step_size = step_size ((2K-1)(2K-1) / ((2K)(2K+1)) x^2)
        value = value - step_size
    return value
```

---

* solving for step size in terms of previous step size - to avoid factorial

$$\sum_{k=0}^{\infty} \frac{(2k)!}{2^{2k}(k!)^2} \cdot \frac{x^{2k+1}}{2k+1}, \quad k(0) = x \qquad = x \cdot \left(\frac{(2k)(2k-1)}{4 \cdot (k^2)}\right) \left(\frac{x^2}{\frac{2k+1}{2k-1}}\right) = \frac{(4k^2 - 2k)x^2}{4k^2\left(\frac{2k+1}{2k-1}\right)}$$

Simplifies to $\frac{(2k-1)(2k-1)}{(2k)(2k+1)} x^2$

$$k = 1: \quad \frac{2!}{2^2(1!)^2} \cdot \frac{x^3}{3} = \frac{2x^3}{12} = \frac{x^3}{6}$$

$$k = 2 \quad \frac{4!}{2^4(2!)^2} \cdot \frac{x^5}{5} = \frac{24x^5}{320} = \frac{3x^5}{40}$$

Credit for simplification:

$$4^k (k!)^2 \cdot (2k+1) = \left[4^{(k-1)} \cdot (k-1)!^2 \cdot (2k-1)\right] \cdot \left[4 \cdot k^2 \left(\frac{2k+1}{2k-1}\right)\right]$$

$$\frac{\pi}{2} - x -$$

ArcTan using Taylor series:

— Note $\arctan(x) = \text{arcCot}\left(\frac{1}{x}\right)$

$$\text{ArcTan}(x) = \arcsin\left(\frac{x}{\sqrt{x^2+1}}\right)$$

ArcSin(x) = approx →

$$x + \left(\frac{1}{2}\right)\frac{x^3}{3} + \left(\frac{3}{8}\right)\frac{x^5}{5} + \left(\frac{1\cdot3\cdot5}{2\cdot4\cdot6}\right)\frac{x^7}{7}$$

$$\text{Arctan}(x) = \arcsin\left(\frac{x}{\sqrt{x^2+1}}\right)$$

↳ will need Sqrt (x) fxn

Ex) input = 5.2, should approach 1.3808

$$\frac{x}{\sqrt{x^2+1}} = 0.982, \quad \arcsin(0.982) \to 1.3808$$

Converges

Taylor approximation **pseudocode** for arcTan:

arctan ( double x ) : ———— * Sqrt fxn needed

plugin_value =(x / Sqrt (x · x + 1)) ———— use tan to sin identity to solve

value = arcsin( plugin_value) ———— call arcsin of plugin_value

return value

**Library Test File:**

Given the array argv, with length argc, first scan the array for which command line inputs the user has given.

Then, first test whether the input a is given, if so, run a helper function to print all outputs. Then, do not test if any of the individual outputs are selected, them running would be re running in all cases.

Otherwise, test if any or all of the other inputs are given.
If this is ever the case, run the helper functions to print the given outputs.

Helper files:
allOutput
sinOuput
cosOutput
tanOutput
logOutput

Library-test uses code (specifically in the getopt stage) inspired from the assignment 2 handout.

General **pseudocode** for helper files:

Int sinOutput():
print output header
print lines
for (int i = -1; i < -1; i+=0.1):

```
        print (i, arcSin, asin, arcSin-asin)
    return 0
```

Note, all helper files will follow this exact format, with all files simply repeating for all individual functions.