Tu-An Nguyen
tunhnguy@ucsc.edu
01/24/2021

CSE 13S Spring 2020
Assignment 2: A Small Numerical Library
Design Document

Purpose

The purpose of this program is to mimic the sine, cosine, tangent, exponent, and logarithmic functions from the `<math.h>` C library. The program will then compare the outputs from the implemented functions to the library's functions.

Planning

1. I will read the Assignment 2 specifications to get a good idea of the subject matter and my task. I will also refer to the supplemental readings outlined at the end of the specifications.
2. I will review some Calculus: derivatives and Taylor series.
3. Before I start coding, I will draft high level pseudocode for each of the functions.
4. I will then start by creating the `Makefile` so I can compile the program. Meanwhile, I can add the build and clean commands to the `README.md` file.
5. I will then start by implementing command-line options into my `main()` function by following the Assignment 2 specifications. I won't worry about implementing the `Sin()`, `Cos()`, `Tan()`, `Exp()`, and `Log()` functions yet, but I will define them so the program can compile and run.
6. Once I get my test harness working, I will begin implementing the `Sin()` and `Cos()` functions.
7. With the completion of the `Sin()` and `Cos()` functions, I can easily implement the `Tan()` function by taking the ratio of sin and cos.
8. The implementation of the `Exp()` function is more confusing to me than the previous ones. I will have to explore how to use an epsilon value to halt computation.
9. To compute log, I will need to use the Newton-Raphson method. This method is new to me, and I will need some time to learn and understand it before implementing it into code. I predict spending the most time on the `Log()` function. I will also need to use my previously implemented `Exp()` function.
10. After sufficient testing, I will begin creating my `WRITEUP.pdf`. I plan to provide graphs to better support my discussion of my test results.

Pseudocode

```
Main:
    o_sin, do_cos, do_tan, do_exp, do_log = false

    while there are options:
        switch:
            case a:
                set all flags to true
                break
            case s:
                do_sin = true
                break
            case c:
                do_cos = true
                break
            case t:
                do_tan = true
                break
            case e:
                do_exp = true
                break
            case l:
                do_log = true
                break
            default:
                print error

    check flags and run respective functions

Sin(x):
    num = x
    den = 1.0
    term = num / den
    sum = term
    for k = 3; |term| > EPSILON; k += 2:
        num *= x * x * -1
        den *= (k - 1) * k
        term = num / den
        sum += term
    return sum
```

```
Cos(x):
    num = 1.0
    den = 1.0
    term = num / den
    sum = term
    for k = 2; |term| > EPSILON; k += 2:
        num *= x * x * -1
        den *= (k - 1) * k
        term = num / den
        sum += term
    return sum

Tan(x):
    return Sin(x)/Cos(x)

Exp(x):
    curr, new, sum = 1.0
    for k = 1.0; |new| > EPSILON; k += 1:
        curr = x / k
        new = new * curr
        sum += new
    return sum

Log(x):
    y = 1.0
    p = Exp(y)
    while |p - x| > EPSILON:
        y += (x - p) / p
        p = Exp(y)
    return y
```

Design Process

1. I read the Assignment 2 specifications.
2. I reviewed calculus including derivatives and Taylor series.
3. Instead of writing pseudocode for all the functions in one go, I decided to only write pseudocode for the `main()` function first. I will then implement it and repeat the process with the next function, working downwards.
    a. While working on `main()`, I referred to section 3 of the Assignment 2 specifications. I ran and studied the provided code to understand the two parameters in the `main()` function: `int argc` and `char **argv`. I also read the

man pages for the `getopt()` function which will be used to parse the arguments given in the command-line.

4. Next, I began writing pseudocode for `Sin()`, `Cos()`, and `Tan()` functions.
    a. The `Sin()` and `Cos()` functions were very similar.
    b. The `Tan()` function only requires the ratio of `Sin()` and `Cos()`.
5. Before implementing the `Sin()`, `Cos()`, and `Tan()` functions, I added a `print_header()` function to print the header of each function's output. I also defined `HEADERFMT` and `NUMFMT` for the functions' output.
6. Then I began implementing the `Sin()`, `Cos()`, and `Tan()` functions.
7. Afterwards, I wrote the pseudocode for the `Exp()` and `Log()` functions.
8. I then implemented the `Exp()` and `Log()` functions.
9. After some testing, I realized I haven't accounted for when `cos(x) == 0` in my `Tan()` function. I decided to test it and the `tan()` function in the `<math.h>` library returns a very large number. I decided to mimic it with my `Tan()` function. I assume this large number is supposed to represent infinity, which would make sense.
10. Lastly, I wrote the `WRITEUP.pdf` and pushed it to my repo.