Numerical Analysis 1 – Class 1

Thursday, January 11th, 2018

Subjects covered

- Class introduction.
- Computer representations of numbers: integers and floating point.
- Computing polynomials, series, and special functions.
- Computer internals.
- Computational error, stability and conditioning.

Readings

- Kutz, Chapter 1 (introduction to Matlab).
- "What every computer scientist should know about floating point", by David Goldberg. Linked on Blackboard.
- "Testing math functions in Microsoft Cloud Numerics", by Brorson, Moskowitz, and Edelman. Linked on Blackboard.

Problems

Most of the following problems require you to write a program. For each program you write, please make sure you also write a test which validates your program. The test should call your function using inputs for which you know the result. The test should then check that your function returns the correct results. You will be graded on both your program as well as on your test. Finally, please place the answers to different questions into different directories and zip up your answers into a single zip file. E-mail your answers to our TA: Mohamed Elbehiry, elbehiry.m@husky.neu.edu.

1. This problem is a programming warm-up exercise. The mathematics is not difficult; the goal is to get used to writing Matlab code. Consider the function Natural Logarithm function for positive real inputs, $\ln(x), x \in \mathbb{R}, x > 0$. This function has several different series expansions; which one you use depends upon the value of *x*. Some series expansions are given at the DLMF under http://dlmf.nist.gov/4.6#i. In particular, consider computing ln(*x*) using the expansions

$$\ln(x) = (x-1) - \frac{1}{2}(x-1)^2 + \frac{1}{3}(x-1)^3 - \cdots, \quad 0 \le x \le 1 \tag{1}$$

$$\ln(x) = \left(\frac{x-1}{x}\right) + \frac{1}{2}\left(\frac{x-1}{x}\right)^2 + \frac{1}{3}\left(\frac{x-1}{x}\right)^3 + \cdots, \quad x > 1/2 \tag{2}$$

Your assignment:

- Write a program to compute the value of ln(*x*) for real inputs $0 < x < 100$ using the above series. To start, select a tolerance of 1e-7 to terminate the summation. Your program should test the input, and depending upon the input select one or the other of the above expressions to compute. Make sure you use Horner's rule for summing series – don't naively compute z^p at each iteration of your loop. (Please explain why not in a README file included in your homework submission.)

- Next write a test program which exercises your program by passing in different values of $0 < x < 100$. Test the value you compute against those returned by the Matlab built-in function log(x). Use a testing tolerance of 1e-6 to compare your results against Matlab's. You may need to adjust the tolerance in your series summation in order to pass your test.
- Now modify your test program to test larger values of x, say x= 1e4. Please explain what happens in the same README file.

2. In problem 1 you used some series expansions to compute ln(x) for positive real x in the domain 0 < x < 100. When you tried using your implementation for larger x you should have found that your implementation either failed tests, or you needed to dramatically increase the number of terms in your sum to get good results. From the perspective of writing correct, performant numerical code, neither of these options is good.

   The problem for large x is that series (2) converges slowly. The way to fix this situation is to use reduction on your input x prior to computing the series expansion. The following algorithm is similar to that used for the ln(x) implementation in fdlibm, which you can see at http://www.netlib.org/fdlibm/e_log.c. It works by dividing the input by 2 while accumulating an additive factor of ln(2) for each division. Then, when the input x is small enough, a series sum for $y = \ln(x)$ is used. The return is then y + n*ln(2), where n is the number of divisions.

```
1. Accept input x and sanity test it (i.e. make sure it's real and positive).
2. Initialize p2 = 0, lg2 = ln(2).
3. Loop:
    4. if x>1.8 then set x = x/2, and set p2 = p2+lg2.
    5. else break out of loop and go to step 7.
6. End of loop.
7. Use series summation valid for 0 < x < 2 to compute y = ln(x).
8. Return y + p2.
```

   Your assignment: Implement this algorithm. Then copy and modify your test program to test this implementation for input x taking values over many orders of magnitude.

3. Write a program to compute the hyperbolic tangent using the definition of the function:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

   As a test, compare your result to the value computed by the Matlab built-in function tanh(x). Note that your program will have problems getting good answers for both large and small input values. Please explain why in a README file handed in with your program. Compare your result for x values in the domain [1e-10, 1e10] by making a plot of the relative error, defined by

$$err = \frac{\tanh_{yours}(x) - \tanh_{matlab}(x)}{\tanh_{matlab}(x)}$$

   Please explain in your README file how you might modify the computation so that it returns better results.