# Numerical Analysis 1 – Class 1

Thursday, January 21st, 2021

## *Subjects covered*

- Class introduction and mechanics.
- How to compute (evaluate) functions:
    - Horner's method. Evaluating polynomials and series expansions.
    - Wallis's' rule. Evaluating continued fractions.
- Computer representations of numbers: integers and floating point.
- Brief overview of 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 Canvas.
- "Testing math functions in Microsoft Cloud Numerics", by Brorson, Moskowitz, and Edelman. Linked on Canvas.
- "Evaluating Continued Fractions … ", by Press, and Teukolsky. Linked on Canvas.

## *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: Hiu Ying Man, man.h@northeastern.edu.

### Problem 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 well-known expression for the number ln(2):

$$\ln(2) = \lim_{N \to \infty} \sum_{k=1}^{N} \frac{1}{k\, 2^k}$$

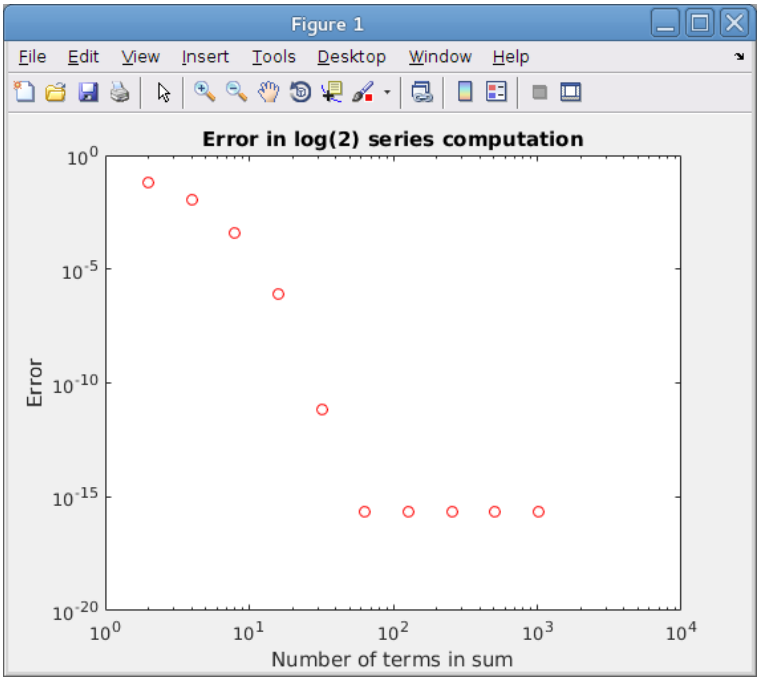
(This expression is given in, for example, "The Quest for Pi", by Bailey et al., available online at https://www.davidhbailey.com//dhbpapers/pi-quest.pdf .)

Your assignment:

1. Write a program to compute this sum for a given input N. In your sum, don't compute $2^k$ with each iteration. Rather, use a variable which is multiplied by 2 each time through your

summation loop. Why do the computation this way?

2. Next write a test program which exercises your program by passing in different values on N and making a plot of your output vs. Matlab's value for ln(2). The result of my test program is shown below. Your result should look similar to mine.
3. The plot of error shows two regimes: a decreasing error for N = [1, 64] and then a flat-line error for N > 64. Please explain what is going on in each regime.




## Problem 2

The natural log function has the following Taylor expansion:

$$\ln(z)=(z-1)-\frac{1}{2}(z-1)^2+\frac{1}{3}(z-1)^3-\ \cdots$$

cf. the DLMF, https://dlmf.nist.gov/4.6 , equation 4.6.3. This expansion is valid for $0<z\leq 2$ . Please write a program to compute the log of an input *z* from this domain. Use Horner's method to implement the series expansion. Stop your summation when the magnitude of the terms in the sum drops below 1e-9. Feel free to use my program arctan_series.m (on Canvas) as a starting point for your code.

To test your implementation, take the result of your computation, $y=mylog(z)$ then compute the round-trip z, $z_{roundtrip}=e^{y}$ . Test the input z against the round-trip z using a tolerance of 1e-8. Test input values in the domain $0.01<z\leq 1.99$ . You my find it necessary to bump up the number of iterations allowed in computing the series in order to achieve convergence over this domain.

## Problem 3

The natural log function has the following continued fraction expansion:

$$\ln(z)=\frac{(z-1)}{1+}\ \frac{1^2(z-1)}{2+}\ \frac{1^2(z-1)}{3+}\ \frac{2^2(z-1)}{4+}\ \frac{2^2(z-1)}{5+}\ \frac{3^2(z-1)}{6+}\ \frac{3^2(z-1)}{7+}\ \frac{4^2(z-1)}{8+}\ \frac{4^2(z-1)}{9+}\ \cdots$$

cf. the DLMF, https://dlmf.nist.gov/4.9 , equation 4.9.1.

Please write a program to compute the log of an input *z* using this continued fraction expression. Use Wallis's algorithm to implement the series expansion. Stop your iteration when the difference between two successive convergents is less than 1e-8. Feel free to use my program computing sqrt(x) as a starting point.

Test your implementation using the same method as used in problem 2. Test the input z against the round-trip z using a tolerance of 1e-8. This time test input values in the domain $0.01 < z \leq 15$ .

### Problem 4

Consider the beta function, defined as

$$B(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1}\, dt$$

Variants of this function appears frequently in many probability distributions. Later in this course we will learn how to evaluate integrals like this one numerically, but for now we will compute this function starting with the well-known identity

$$B(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$$

Since the $\Gamma$ function is available in Matlab as gamma(x), the naive approach to compute B(x, y) is to simply evaluate this equation. This approach has problems, as we shall see.

Please do the following:

- Write a program called mybeta_naive, which computes B(x, y) using the identity above.
- Write a test function which calls mybeta_naive, and compares the return to that computed by Matlab. Do the computation for x, y inputs drawn from the set $x, y \in [0.1, 1, 10, 100]$ . What problem do you find?
- Besides gamma(x), Matlab also supplies the function gammaln(x) = ln(G(x)). This function is better behaved for large and small input arguments. Please use this function to write a second program called mybeta_gammaln which returns values for B(x, y). Test it using your test function and compare the results against those returned by mybeta_naive. Which of the two computations is better behaved for large inputs?