Numerical Analysis 1 – Class 1

Friday, January 13th, 2023

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 Pell numbers. This is a sequence of numbers defined by the recursion relation

$$p_n = 2\,p_{n-1} + p_{n-2} \qquad (1)$$

with seed conditions $p_0 = 0$ and $p_1 = 1$. (You will note the Pell numbers are analogous to their more-famous cousins, the Fibonacci numbers.) You can read about the Pell numbers at Wolfram MathWorld, https://mathworld.wolfram.com/PellNumber.html.

Please do the following:

- Write a program which takes as input an integer N, and returns the first N Pell numbers stored in a vector. This is an easy exercise in writing "for" loops.

- Write a test program for your implementation. Your test program will call your implementation with, say, *N*=100, and then test that the returned vector of numbers is correct. Please implement the following two tests:
  - Please check that each returned number satisfies the recurrence relation (1) above.
  - Please check that each returned number satisfies the "Binet-type" formula

$$p_n = \frac{(1+\sqrt{2})^n - (1+\sqrt{2})^n}{2\sqrt{2}}$$

  You will find a naive "Binet-type" test will fail for larger p_n. Please explain why. Hint: You can fix the failing tests by using a relative (instead of an absolute) tolerance for your testing.

Problem 2

In this problem you will compute the function tanh(x) using a two-tier approach. The idea is similar to that used to compute sin(x) which was presented in class.

First, consider the Taylor expansion for tanh(x) given in the DLMF (equation 4.33.3),

$$\tanh(x) = x - \frac{1}{3}x^3 + \frac{2}{15}x^5 - \frac{17}{315}x^7 + \ldots + \frac{2^{2n}(2^{2n}-1)B_{2n}}{(2n)!}x^{2n-1} \qquad (2)$$

This expansion looks complicated, but all pieces are readily computed in Matlab except the coefficients B_{2n}. The B_{2n} are the Bernoulli numbers which were originally discovered in the context of summing powers of integers. If you're interested you can read more about them on the web, but for this problem you just need to consider them as numerical coefficients in your expansion.

Computing the Bernoulli numbers accurately is difficult for larger *n*, so I have placed a very simple program called "mybernoulli.m" onto Canvas which returns the first 34 Bernoulli numbers. Please use that program for your work.

Please do the following:

- Write a program which computes the series expansion (2) above using Horner's method. Since you have only the first 34 Bernoulli numbers, your program should loop a maximum of 34 times. Restrict the input domain to real numbers $|x| < \pi/2$ since that is the convergence domain of (2).
- Write a program to test your series implementation. You should find that your tests fail for input values $|x| \gtrsim 1$, depending upon your testing tolerance. (I used 1e-6 as my tolerance.) This happens because the series expansion requires more terms as the input *x* gets closer to the end of the convergence domain.
- The bad convergence for $|x| \gtrsim 1$ presents a problem: We only have access to the first 34 Bernoulli numbers so we can't add more terms to the series to get better accuracy. How to get more accuracy so the tests will pass? One answer is to use the identity (DLMF 4.35.3),

$$\tanh(u+v) = \frac{\tanh(u) + \tanh(v)}{1 + \tanh(u)\tanh(v)} \qquad (3)$$

  Knowing this identity, write a new program which does this:
  - Examine the input *x*. If $|x| < \text{threshold}$ then just call your series implementation

created above.

  - If $|x| > \text{threshold}$ then split the input into two pieces, $x = u + v$ where both $|u| < \text{threshold}$ and $|v| < \text{threshold}$.
  - Now call the series expansion on each of the two pieces, *u* and *v*. Then create the final return value using the identity (3).
  - Regarding what value to use for the threshold, you might need to experiment a little bit, but it is clear that the threshold should be close to – but less than – one.
- Now write a second test program for your new implementation. You should find that all tests created by this program will pass.

When you are done with this problem you will have created a two-level program for computing tanh(x) where the upper level reduces the input value and the lower level program computes a series expansion. This is a reasonable strategy for computing many functions where no approximation works well over the entire input domain of the function.

Problem 3

In this problem you will compute tanh(x) using a continued fraction expansion. The expansion to use is (DLMF 4.39.1)

$$\tanh(x) = \cfrac{x}{1 + \cfrac{x^2}{3 + \cfrac{x^2}{5 + \cfrac{x^2}{7 + \ldots}}}} \tag{4}$$

Please do the following:

- Write a program using the Wallis algorithm shown in class to compute tanh(x) from the expansion (4). Note that this continued fraction's convergence domain is the entire complex plane except near a countably-infinite number of discrete poles. Nonetheless, go ahead and restrict the inputs to real numbers in the domain $|x| < \pi/2$.
- Write a program which tests your continued fraction implementation. You should find that the continued fraction expansion converges quickly for the entire $|x| < \pi/2$ input domain.

Problem 4

This simple problem invites you to examine the pitfalls of computing with floating point numbers. Consider the polynomial $p(x) = (x-1)^7$. Please do the following:

- Write a program which plots this polynomial over the domain $x \in [1-\delta, 1+\delta]$ where $\delta = 1.2\text{e-}2$. Note that in the vicinity of $x \approx 1$ the value of $p(x)$ will be very small.
- Write down the expression obtained when expanding this polynomial into a sum of monomials. (i.e. write down the binomial expansion for $(x-1)^7$) Rather than multiplying everything out by hand, just recall the fact that the coefficients in the expansion are rows of Pascal's triangle.
- Modify the program you just wrote to plot the expanded polynomial on the same plot. My result is shown below.
- Please answer the question, what is going on? Be as specific as possible. This is a pencil and

paper exercise – please turn in your written answer.

There is no test needed for this exercise. Just turn in your program.

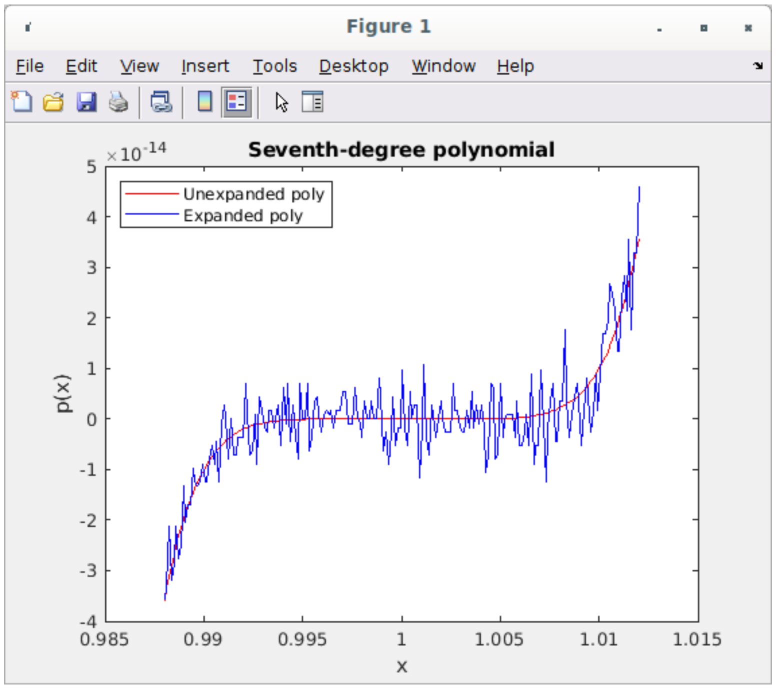
Figure 1
File Edit View Insert Tools Desktop Window Help
Seventh-degree polynomial
×10^-14
Unexpanded poly
Expanded poly
p(x)
x