Reading: Read the syllabus and ‘Guidelines for Code’ document on Piazza. Make sure you understand the expectations for homework.

Also read Chapter 1.1 of Moler’s book (linked from the syllabus). If using python, you may want to review a tutorial for numpy (e.g. the ones linked on Piazza) if you are not already familiar with it. Whichever language you plan to use, I suggest taking a look at the other.

Note: We will not have need of the syms command or any of the symbolic toolbox (which is referenced sometimes in Moler’s book), so you can ignore it when mentioned.

Other work: Download the plotting example (Matlab or python) from Piazza, read through it and run it. If using python, you will need to have the right packages installed.

Fill out the Office Hours Poll on Piazza (office hours will be scheduled next week).

Problems

P1 (open-ended). Briefly describe an area of interest where numerical computation is relevant. What kinds of mathematical problems are involved? What practical concerns, if any, are relevant (efficiency, accuracy and so on)?

Note: You could, instead, consider some problem you are interested in knowing how to solve numerically (it does not have to be a real-world application).

P2. a) Consider a floating point system with a given $\beta$ and $t$ and let $\eta = \frac{1}{2}\beta^{1-t}$ be the rounding unit. Explain why

$$\text{fl}(1 + x) = 1$$

if $0 < x < \eta$,

assuming that the fl operation rounds to the nearest floating point number and that $1 + x$ is evaluated exactly.

b) Test this on a computer for a double (with $\eta = 2^{-53}$) by computing $1 + 2^{-k}$ for $k = 0, 1, 2, \cdots$ (you don’t need to show any work here).

c) We saw in class that the formula

$$f'(1) \approx \frac{f(1 + h) - f(1)}{h}$$

has a rounding error becomes a problem when $h$ is small. What happens when $0 < h < \eta$?

---

1Note: the default non-integer in Matlab and python is a double. In python, if you type an integer, e.g. $x=1$ it will set $x$ to be an integer and not a double, but $a=1.0$ will make $x$ a double.
**P3.** In this problem we will see how rearranging a calculation can make it more efficient. Consider the problem of evaluating an $n$-th degree polynomial

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

at a point $x$.

a) The algorithm below uses the formula (1) to compute $P_n(x)$. How many operations (multiplications and additions) are required?

**Algorithm 1** Naïve polynomial evaluation

**Input:** $n \geq 0, x \in \mathbb{R}$ and $\{a_0, a_1, \ldots, a_n\}$

**Output:** $y = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

```plaintext
y ← a_0
z ← x ▷ stores $x^i$
for $i = 1, \ldots, n-1, n$ do
    y ← y + za_i
    z ← xz
end for
return y
```

b) A better approach is **Horner’s method**, which proceeds by writing

$$P_n(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n)) \cdots).$$

For example, a quadratic would be evaluated as

$$ax^2 + bx + c = c + x(b + ax).$$

Write an algorithm (in pseudocode, as above, or actual code if you prefer) for calculating $P_n(x)$ using Horner’s method. **Hint:** start with $xa_n$, then add $a_{n-1}$, etc. How many operations are required?