18.335 Problem Set 1

There are four problems below (Problem 1 - Problem 4). You may complete as many problems as you want, but please submit **only three** problems for grading. **Due March 2, 2025 at 11:59pm.** Late submission will only be accepted (with mild penalty) up to **three days** after the due date. You should submit your problem set **electronically** on the 18.335 Gradescope page. Submit **both** a *scan* of any handwritten solutions (I recommend an app like TinyScanner or similar to create a good-quality black-and-white "thresholded" scan) and **also** a *PDF printout* of the Julia notebook of your computer solutions. A **template Julia notebook is posted** in the 18.335 web site to help you get started.

`Problem 0: Pset Honor Code`

Include the following statement in your solutions:

> *I will not look at 18.335 pset solutions from previous semesters. I may discuss problems with my classmates or others, but I will write up my solutions on my own. <your signature>*

Problem 0.1: Jupyter notebook

On the course home page, "launch a Julia environment in the cloud" and open the "Floating-Point-Intro.ipynb" notebook in the notes folder. Read through it, get familiar with Julia and the notebook environment, and play! (You don't need to submit a notebook print out or turn in work for this question.)

Problem 1: Floating point

The nonzero floating point numbers $\mathbb{F}$ can be written compactly in the form (e.g., see (13.2) in Trefethen and Bau) $x = \pm(m/\beta^t)\beta^e$, with integer base $\beta \geq 2$, significand $\beta^{-1} \leq m/\beta^t < 1$, and integer exponent e.

(a) This floating point system includes many integers, but not all of them. Give an exact formula for the smallest positive integer n that does not belong to $\mathbb{F}$, and explain how you get the formula. In particular, what are the values of n for IEEE single and double precision arithmetic? Figure out a way to verify this result for your own computer. (You may use Julia, which employs IEEE double precision by default. However, unlike Matlab, Julia distinguishes between integer and floating-point scalars. For example, `2^50` in Julia will produce a 64-bit integer result; to get a 64-bit/double floating-point result, do e.g. `2.0^50` instead.)

(b) Consider two floating numbers a, b with $a \leq b$. Prove that in base 2 arithmetic, the following inequality holds:

$$a \leq fl((a+b)/2) \leq b$$

Here, $fl(\cdot)$ maps a real number to the nearest floating point number. However, this property does not necessarily hold in base-10 floating-point arithmetic. Construct a counterexample demonstrating a violation in base-10 arithmetic and verify it on your own computer. (You may want to use the DecFP package to perform decimal floating-point arithmetic in Julia)

Problem 2: Rounding error analysis

In class, we analyzed the rounding error of the naive inner product. In this problem, we will examine the rounding error in different implementations of the summation function.

Method 1: Let $\hat{f}(x)$ be the naive summation function that sums the numbers from left to right. The naive summation can be generalized by breaking the sum into k pieces, with each mini sum of length n/k being evaluated separately. The final sum is then obtained by summing these intermediate results:

$$\bar{f}(x) = \hat{f}(s_1, \ldots, s_{\lceil n/k \rceil}),$$

where $s_i = \hat{f}(x_{(i-1)k+1}, \ldots, x_{\min\{ik,n\}})$. Here, $\lceil y \rceil$ denotes the smallest integer $\geq y$ (i.e. y rounded up).

Method 2: Compute $\tilde{f}(x)$ by a recursive divide-and-conquer approach, recursively dividing the set of values to be summed in two halves and then summing the halves:

$$\tilde{f}(x) = \begin{cases} 0 & \text{if } n = 0 \\ x_1 & \text{if } n = 1 \\ \hat{f}(\tilde{f}(x_{1:\lfloor n/2 \rfloor}), \tilde{f}(x_{\lfloor n/2 \rfloor + 1:n})) & \text{if } n > 1 \end{cases},$$

where $\lfloor y \rfloor$ denotes the greatest integer $\leq y$ (i.e. y rounded down).

(a) Perform an error analysis for the blocked summation method. For which choice of k can we minimize the error's dependence on n?

(b) Analyze the rounding error for Method 2 and show that the worst-case error bound grows logarithmically rather than linearly in n. In the pset 1 Julia notebook, there is a function "div2sum" that computes $\tilde{f}(x)$. Plot the rounding error for random inputs as a function of n. Are your results consistent with your error bound?

Problem 3: Matrix norm

(a) Derive Trefethen eq. (3.10): for $A \in \mathbb{C}^{m\times n}$, the subordinate ∞-norm is equal to the "maximum row sum,"

$$\|A\|_\infty = \max_{1\le i\le m} \|a_i^*\|_1 \,,$$

where a_i^* denotes the ith row of A. Find the code that computes the induced $\|A\|_\infty$ norm in Julia, the `opnorm(A, Inf)` function, on `https://github.com/JuliaLang/LinearAlgebra.jl` in src/generic.jl and satisfy yourself that it is equivalent to (3.10).

(b) Let $\|\cdot\|$ be a subordinate norm. Suppose $B \in \mathbb{R}^{n\times n}$ satisfies $\|B\| < 1$. Show that the matrix $I + B$ is nonsingular, and its inverse satisfies the bound

$$\|(I+B)^{-1}\| \le \frac{1}{1-\|B\|} \,.$$

Problem 4: SVD and applications

(a) Suppose $A \in \mathbb{C}^{n\times n}$ has singular values $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n > 0$. Let $\alpha \in \mathbb{C}$ be a chosen such that the following matrix is nonsingular:

$$C = \begin{bmatrix} \alpha I & A^* \\ A & \alpha I \end{bmatrix} .$$

Compute the 2-norm $\|C\|$, and the 2-condition number $\kappa_2(C)$. Check your answer on your own computer.

(b) The *stable rank* of a matrix $A \in \mathbb{R}^{n\times n}$ is defined as

$$\operatorname{srank}(A) = \frac{\|A\|_{\mathrm{F}}^2}{\|A\|_2^2}$$

It serves as a useful proxy for the rank of A. Show that the stable rank satisfies the bounds $1 \le \operatorname{srank}(A) \le \operatorname{rank}(A)$. Check the bound for a random matrix on your own computer.

Feedback (optional)

Please let me know how you're finding the course and the first problem set. What are you hoping to get out of the class? How is the pace of lecture? Please rate the difficulty and volume of the first problem set. You can submit an anonymous survey here.