18.335 Problem Set 2

Due Fri. Mar 12, 2021, at 3pm via Stellar.

Problem 1: Stability

(a) Trefethen, exercise 15.1. You will need to review the definition of "stability," of which backwards stability is a (common) special case, in that chapter. [In parts (e) and (f), assume that $\frac{1}{k!}$ can be computed to $O(\epsilon_{\text{machine}})$ and concentrate on the accumulation of errors in the summations.]

Note: in part 15.1(c), the online version of Trefethen has a misprint: it should be $\tilde{f}(x) = x \oslash x$ as an algorithm for $f(x) = 1$, **not** "$\tilde{f}(x) = x$."

(b) Trefethen, exercise 16.1.

Hint: for $f(A) = QA$ where Q is unitary, it might be convenient to show that backwards stability of $\tilde{f}(A)$ **in this case only** is equivalent to proving $\|\tilde{f}(A) - f(A)\| = \|f(A)\| O(\epsilon_{\text{machine}})$ if you choose $\|\cdot\|$ to be the L_2 induced norm or the Frobenius norm. i.e. for multiplying by a *unitary* matrix, backwards stability is equivalent to the *forwards* error being small, which might be easier to analyze. (Essentially, this happens because unitary matrices have condition number $\kappa(Q) = 1$.) Once you have proved stability for multiplying by *one* Q, you can prove backwards stability for multiplying by *many* Q's using induction, for example.

Problem 2: Norms

(a) Derive Trefethen eq. (3.10) (for which Trefethen only writes "by much the same argument"). Find the code that computes the induced $\|A\|_\infty$ norm in Julia, the `opnorm(A, Inf)` function, on `github.com/JuliaLang/julia` in stdlib/LinearAlgebra/src/generic.jl and satisfy yourself that it is equivalent to (3.10).

(b) Trefethen, problem 3.4. Check your result for a random 10×7 matrix A in Julia, constructed by `A=randn(10,7)` with the induced $p = 2$ norm as computed by `opnorm(A)` in Julia.

Problem 3: SVD and low-rank approximations

(a) Trefethen, probem 4.5.

(b) Trefethen, problem 5.2.

(c) Trefethen, problem 5.4.

Problem 4: Least squares

Trefethen, problem 11.2. Note that the $\Gamma(x)$ function is provided as `gamma(x)` by the SpecialFunctions package in Julia (execute `] add SpecialFunctions` to install this package). You might also want to google the "Laurent series" for the gamma function.

Note that the L^2 norm $\|g(x)\|_2$ of a function $g(x)$ defined on $x \in [a, b]$ is an *integral*

$$\|g(x)\|_2 = \sqrt{\int_a^b |g(x)|^2 dx}.$$

On a computer, you will need to approximate such integrals by a finite *sum* over N points with some weights, which will turn this fitting problem into an ordinary least-squared matrix problem.

Such an approximation is called a "quadrature" rule, and we will study quadrature in more detail later, but for now you can use whatever simple approximation you like—the simplest is probably a "rectangle" rule or "Riemann sum" (google it), and you probably saw something like it the first time you learned about integration. As you increase N (for any quadrature rule), your sum should get closer and closer to the integral, and you should keep doubling N until your final answer(s) converge to at least 2 significant digits.