18.335 Problem Set 2

Due Wed. Feb 26, 2020, 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.]

(b) Trefethen, exercise 16.1.

(c) Consider the function $f(x, y) = x^* y = \sum_{i=1}^{n} \overline{x_i} y_i$ for $x, y \in \mathbb{C}^n$, computed by naive (left-to-right) summation of the floating-point products $\overline{x_i} \otimes y_i$. Show that this algorithm is backwards stable (very similar to the proof of backwards stability of summation from class). Here, the inputs (x, y) live in the *Cartesian product* space $\mathbb{C}^n \times \mathbb{C}^n$, which is a vector space using the obvious operations $(x, y) + (z, w) = (x + z, y + w)$ and $\alpha(x, y) = (\alpha x, \alpha y)$. You will need to choose a convenient norm of $\mathbb{C}^n \times \mathbb{C}^n$ to apply the backwards-stability definition.

(Food for thought: Your solution could easily be adapted to establish backwards stability of matrix products $f(A, B) = AB$ computed by the familiar "row · column" algorithm. Not required here, though.)

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).