18.335 Problem Set 3

Due Friday, 18 March 2022 at 12pm.

Problem 1: QR and RQ

(a) Trefethen, problem 10.4.

(b) Trefethen, problem 28.2. In the second part, you *must* additionally assume that $A = A^*$ (i.e. it is Hermitian tridiagonal), as otherwise (for non-Hermitian tridiagonal A) RQ would *not* be tridiagonal. (Some editions of the book omitted this requirement.)

Problem 2: Distribution and association

Suppose you want to compute $x^T(AB + CD)y$, where $x, y \in \mathbb{R}^m$ and $A, B, C, D \in \mathbb{R}^{m \times m}$ for $m = 1000$. You code it up in Julia in two ways:

- `x' * (A*B + C*D) * y`
- `x'*A*B*y + x'*C*D*y`

(a) Which of these two would you expect to be faster, and why? (Note that $*$ in Julia is "left-associative:" performed from left to right, unless you change the order with parentheses.)

(b) Try it and see if it maches your prediction.

A good package for benchmarking in Julia is BenchmarkTools.jl — install it with `] add BenchmarkTools`, load it with `using BenchmarkTools`, allocate random inputs and time them with e.g. `@btime $x' * ($A*$B + $C*$D) * $y`; the $ signs tell the benchmark to evaluate the global variables like `x` before benchmarking to avoid an artificial slowdown (global variables are otherwise slow in Julia).

Problem 3: 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-squares matrix problem. Such an approximation is called a "quadrature" rule: 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.

Problem 4: Schur factorization

In class, we will discuss the *Schur factorization*: any square $m \times m$ matrix A can be factored as $A = QTQ^*$, where Q is unitary and T is an upper-triangular matrix (with the same eigenvalues as A, since the two matrices are similar).

(a) A is called "normal" if $AA^* = A^*A$. Show that this implies $TT^* = T^*T$. From this, show that T must be diagonal. Hence, any normal matrix (e.g. unitary or Hermitian matrices) must be unitarily diagonalizable. Discuss the connection between this result and the SVD of A.

(b) Given the Schur factorization of an arbitary A (not necessarily normal), describe an algorithm to find the eigenvalues and eigenvectors of A, assuming for simplicity that all the eigenvalues are distinct. The flop count (count of real $\pm, \times, \div$; assume that your matrices A, T, Q are all real for simplicity) should be asymptotically $Km^3 + O(m^2)$; give the constant K.