18.335 Take-Home Midterm Exam: Spring 2021

Posted 3pm Thursday April 15, due **3pm Friday April 16.**

Problem 0: Honor code

Copy and sign the following in your solutions:

I have not used any resources to complete this exam other than my own 18.335 notes, the textbook, running my own Julia code, and posted 18.335 course materials.

_____

your signature

Problem 1: (34 points)

Horner's method evaluates the degree $n-1$ polynomial $p(x) = c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}$ via

$$p(x) = c_0 + x(c_1 + x(c_2 + x(\cdots)))$$

using $n-1$ additions and $n-1$ multiplications.

Define

$$f(c,x) = c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}$$

where $c \in \mathbb{R}^n$ are the n coefficients and $x \in \mathbb{R}$ is the x value. Define $\tilde{f}(c,x)$ to be the algorithm that evaluates $f(c,x)$ in floating-point arithmetic (replace addition with $\oplus$ and multiplication with $\otimes$) by Horner's method.

- Show that $\tilde{f}$ is **backwards stable** with respect to the input c by itself, for any given x, in the usual sense from class: $\tilde{f}(c,x) = f(\tilde{c},x)$ for some $\tilde{c}$ satisfying $\|\tilde{c}-c\| = \|c\|O(\varepsilon_{\text{machine}})$ (the leading coefficient in the $O(\varepsilon)$ can depend on n and x but not c). For simplicity, you can assume that $c \in \mathbb{F}^n$ and $x \in \mathbb{F}$.
  - Hint: you can write Horner's method as a recurrence: define $s_k = c_k + xs_{k+1}$ with $s_{n-1} = c_{n-1}$, so that $f = s_0$ (and more generally $s_k = \sum_{j=k}^{n-1} c_j x^{j-k}$). Similarly for $\tilde{f}$.
  - Note also that you can bound polynomials like $p(x)$ as $|p(x)| \le \|c\|_1 \cdot \max\{1, |x|^{n-1}\}$ in terms of $\|c\|$.

Problem 2: (33 points)

Suppose that

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \overline{\beta_1} & \alpha_2 & \beta_2 & & \\ & \overline{\beta_2} & \ddots & \ddots & \\ & & \ddots & \alpha_{m-1} & \beta_{m-1} \\ & & & \overline{\beta_{m-1}} & \alpha_m \end{pmatrix}$$

is an $m \times m$ Hermitian ($T = T^*$) complex tridiagonal matrix.

(a) Show that $T = D\hat{T}D^{-1}$ where $\hat{T}$ is a *real* tridiagonal matrix and D is *diagonal and unitary.*

(b) In computing the eigenvalues of T, istead of applying QR iterations directly to T, you should instead apply QR to _____, which gives the same result because _____ and is faster (in terms of flop counts) because _____.

Problem 3: (33 points)

Suppose A is an $m \times n$ matrix with $n > m$ and rank m (linearly independent rows): a "wide" matrix. In this case, $Ax = b$ has infinitely many solutions x for any right-hand side $b \in \mathbb{C}^m$ (it is an *underdetermined* system of equations). We compute the QR factorization of the conjugate-transpose $A^* = QR = \hat{Q}\hat{R}$ by some backwards-stable algorithm (e.g. Householder QR), where Q is $n \times n$ and R is $n \times m$, and $\hat{Q}$ is the "thin" QR (the first m columns of Q) and $\hat{R}$ is correspondingly the first m rows of R.

(a) Using this QR factorization, devise an efficient algorithm to find the *minimum-norm* solution x of $Ax = b$: the solution x for which $\|x\|_2$ is as small as possible.

- Hint: write the solutions x in the basis of the columns of $Q = \begin{pmatrix} \hat{Q} & Q_\perp \end{pmatrix}$, where the $n - m$ columns $Q_\perp$ span the orthogonal complement of $\hat{Q}$. In this basis, which components are the same for all solutions of $Ax = b$, and which can vary? What does that tell you about the minimum-norm solution in this basis?

(b) What is the cost of your algorithm in the "big O" sense (i.e. is it $\Theta(m^3n^2)$ or ...?), *not* including the $\Theta(nm^2)$ cost of finding the QR factorization by Householder?