18.335 Midterm Exam: Spring 2019

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, and posted 18.335 course materials.

your signature

Problem 1:

If the matrix A is a subset of the columns of matrix B, show that $\kappa(A) \leq \kappa(B)$, where κ denotes the condition number of the matrices. (Note that since A is non-square, you can't use the $\|A\| \cdot \|A^{-1}\|$ definition of $\kappa(A)$ but must use instead the more general definition from the upper bound of equation 12.9 in the book.)

Problem 2:

For a diagonalizable $m \times m$ matrix $A = X\Lambda X^{-1}$, the matrix square root is

$$A^{\frac{1}{2}} = X\Lambda^{\frac{1}{2}}X^{-1} = X \begin{pmatrix} \sqrt{\lambda_1} & & & \\ & \sqrt{\lambda_2} & & \\ & & \ddots & \\ & & & \sqrt{\lambda_m} \end{pmatrix} X^{-1}.$$

(a) For general matrices, the $X\Lambda^{\frac{1}{2}}X^{-1}$ may not be accurate because ....? (One-sentence answer, please.)

(b) Your friend Alyssa P. Hacker is using the $X\Lambda^{\frac{1}{2}}X^{-1}$ formula, but is not worried about accuracy because she can see by inspection (no calculation required) that her A matrices (which are not sparse) are all .... (Give a good example of a valid reason.)

Problem 3:

Suppose that you have a vector $x \in \mathbb{F}^n$ of n double-precision floating-point values (but no ±Inf or NaN) and you want to compute

$$f(x) = \log\left(\sum_{i=1}^{n} e^{x_i}\right)$$

accurately. [You are given the usual library functions that compute log (of positive values) and exp accurately for individual $\mathbb{F}$ inputs (to a forward relative error of a few times $\varepsilon_{\text{machine}}$, i.e. to within a few ulps).] Your friend J. Harvard wrote some code directly from the definition, above, but you notice that it gives Inf for a lot of inputs. Explain how to fix this problem: describe an algorithm that will get a reasonably accurate answer for arbitrary x. (You don't need to prove backwards stability.)

Problem 4:

If we have good initial guesses for one or more of the eigenvalues of the $m \times m$ **Hermitian** matrix $A = A^*$ then we can simply apply Newton's method to find a root of $f(z) = \det(A - zI)$. However,

(a) We don't want to explicitly form the polynomial $f(z)$ in terms of its coefficients, since we saw in class and in the book that any tiny error in the coefficients can lead to a huge error in the roots.

(b) Evaluating $f(z)$ by doing the LU factorization of $A - zI$ for each z (and then multiplying the diagonal entries of U) would be too slow, $\Theta(m^3)$ for every z.

You are allowed do $\Theta(m^3)$ preprocessing steps *once* on the matrix A to compute some factorization of your choice (but *not* its diagonalization or Schur form—your preprocessing must be exact in exact arithmetic, not an iterative method like QR$\longrightarrow$RQ). Given that factorization of A, describe (in pseudocode) an $O(m)$ algorithm to compute $f(z)$ for any z, enabling a fast Newton's method.

- Newton's method would also require $f'(z)$, which could easily be obtained from your fast $f(z)$ algorithm, but you don't need to supply this algorithm.
- Don't worry about overflow/underflow. (In practice, this is easily avoided since Newton's method only needs the ratio f/f'.)