Numerical analysis: Homework 3

Instructor: Anil Damle

Due: February 28, 2025

Policies

You may discuss the homework problems freely with other students, but please refrain from looking at their code or writeups (or sharing your own). Ultimately, you must implement your own code and write up your own solution to be turned in. Your solution, including plots and requested output from your code should be typeset and submitted via the Gradescope as a pdf file. This file must be self contained for grading. Additionally, please submit any code written for the assignment as zip file to the separate Gradescope assignment for code.

Question 1:

Consider the following $n \times n$ matrix

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 1 \\ -1 & 1 & 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ -1 & \cdots & -1 & 1 & 0 & 1 \\ -1 & \cdots & -1 & -1 & 1 & 1 \\ -1 & \cdots & -1 & -1 & -1 & 1 \end{bmatrix}$$

that where all the diagonal and last column are all 1, all the entries below the diagonal are -1, and all other entries are zero.

(a) Consider computing an LU decomposition of A (with or without partial pivoting, the answer will be the same either way if we break ties when pivoting by using the smallest possible index). Work out an expression for the largest entry of U in terms of n.

(b) If we computed an LU factorization in practice and encountered an entry of this size do you think it would be problematic? why or why not?

(c) Does the problem persist if we use complete pivoting?

Note that this matrix demonstrates the worst case behavior for the size of elements in U when computing an LU factorization with partial pivoting. Nevertheless, such cases are considered rare in practice and LU with partial pivoting is widely used.

Question 2:

To balance the prior question, we will consider LU with partial pivoting applied to random problems. (Admittedly, while application driven problems likely don't mirror the pathological worst case discussed above they are also not random.)

For this problem you may either implement LU with partial pivoting yourself, or use a built in routine (make sure you read the documentation and are calling it correctly). We will also use a slight simplification of the growth factor to facilitate using a built in routine and define ρ as

$$\rho = \frac{\max_{i,j}|u_{ij}|}{\max_{i,j}|a_{ij}|}$$

where $PA = LU$.

(a) Build a sequence of random $n \times n$ matrices A of increasing size (starting at $n = 100$ and reaching at least $n = 1000$) and for each one compute its partially pivoted LU factorization. Here, random means having independent and identically distributed entries sampled from $\mathcal{N}(0, 1)$. Using logarithmic axes plot ρ vs n for your test matrices. What do you observe? Empirically, how does the growth factor grow with n?

(b) Using the same style of random matrices as above, for $n = 100$ generate at least 1000 random A, compute their partially pivoted LU factorization and record the growth factor. Plot the empirical distribution of the growth factor (e.g., via an appropriate histogram). What do you observe?

Question 3:

Consider a symmetric positive definite matrix A that is tridiagonal, i.e., $A_{i,j} = 0$ if $|i-j| > 1$. Devise a scheme to compute the Cholesky factorization of A using $\mathcal{O}(n)$ arithmetic operations. (Ideally it would also be $\mathcal{O}(n)$ storage assuming we are given A that way, but we will focus on arithmetic complexity here.) Provide an unambiguous description of your algorithm, e.g., via pseudocode, prove it has the desired complexity, and prove it computes the desired factorization.

Question 4:

Consider the least squares problem

$$\min_x \|b - Ax\|_2$$

where $A \in \mathbb{R}^{m \times n}$ is of rank n, $m > n$, and $b \in \mathbb{R}^m$. Assume someone computed a reduced QR factorization in exact arithmetic using Householder reflectors of the $m \times (n+1)$ matrix $\begin{bmatrix} A & b \end{bmatrix}$

$$\begin{bmatrix} A & b \end{bmatrix} = QR$$

where $Q \in \mathbb{R}^{m \times (n+1)}$ and $R \in \mathbb{R}^{(n+1) \times (n+1)}$. Now assume that whomever computed this QR factorization misplaced Q (or any way to represent it), A, and b leaving you only with R. Nevertheless, they would like for you to find the solution to the underlying least squares problem.

Answer the following using only R. (The preceding story was my long winded way of saying that you may not use b, A, or Q for any purpose in your algorithms for the following problems. You may use their existence when proving why your methods return the correct result).

(a) Prove that you can use a single logical operation to definitively determine whether or not R is singular. You may not use any arithmetic operations in this problem. (More precisely, you are allowed to perform a single operation of the form $<$, $\leq$, $=$, $\geq$, or $>$ on any two numbers of your choosing that are readily available to you, such as entries of R and constants. Based on the result of your logical operation you have to state, correctly, whether or not R is singular.)

(b) Devise an algorithm to determine the 2-norm of the residual $b - Ax$ at the solution to the least squares problem in $\mathcal{O}(1)$ time. Be sure to carefully show why your algorithm is returning the desired result.

(c) Devise an algorithm to determine the solution x of the least squares problem in $\mathcal{O}(n^2)$ time. Be sure to carefully show why your algorithm is returning the desired result.

Turns out a QR factorization of $\begin{bmatrix} A & b \end{bmatrix}$ *is a reasonable (in fact, good) way go when we want to solve a least squares problem.*