CS 4220 / MATH 4260: Homework 2

Instructor: Anil Damle
Due: February 23, 2018

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 CMS as a pdf file. Additionally, please submit any code written for the assignment via the CMS as well. This can be done by either including it in your solution as an appendix, or uploading it as a zip file via the CMS.

Question 1:

When discussing Gaussian elimination and then LU decompositions in class, we considered matrices $G^{(k)}$ that when applied to a matrix A introduced zeros in the k^{th} row below the diagonal. Assuming all of the matrices involved in this problem are $n \times n$, we may write these matrices compactly as

$$G^{(k)} = I - \ell^{(k)} e_k^T,$$

where $\ell^{(k)} = [0, \ldots, 0, \ell_{k+1,k}, \ldots \ell_{n,k}]^T$ is zero in the first k entries. We will now prove two properties of these matrices outlined in class.

(a) Prove that $\left(G^{(k)}\right)^{-1} = I + \ell^{(k)} e_k^T$

(b) Prove that $\left(G^{(1)}\right)^{-1} \left(G^{(2)}\right)^{-1} \cdots \left(G^{(n-1)}\right)^{-1} = I + \sum_{k=1}^{n-1} \ell^{(k)} e_k^T$

Question 2:

Assume that you are given an $n \times n$ non-singular matrix of the form

$$A = D + u e_n^T + e_n v^T$$

where u and v are length n vectors, and D is a diagonal matrix whose diagonal entries we will encode in the vector d, *i.e.* $D_{i,i} = d_i$. (In the course of thinking about this problem, you may stumble across the so-called Sherman-Morrison-Woodbury formula and find that mathematically it provides a potential solution—you are welcome to try it out and see what happens. However, it will yield an even worse solution, both in terms of accuracy and the residual, for the problem instance in part (c) than working off of an LU factorization (even though we have omitted pivoting in this case).

(a) If we assume that we may compute the LU decomposition of A without any pivoting (**note that in general this is not a good idea, but for the purpose of this question we will make such an assumption**), devise a means to solve a linear system of the form $Ax = b$ using $\mathcal{O}(n)$ time. Here we will also consider the cost of memory allocation and therefore you cannot form A explicitly as that would take $\mathcal{O}(n^2)$ time.

(b) Implement your method and demonstrate that it achieves the desired scaling.

(c) Download the files on the course website containing vectors d, v, b and the true solution x_{true}. Solve the associated linear system yourself for an x that satisfies $Ax = b$. Using a logarithmic scale for the error, plot both the absolute error of the difference between your computed solution and x_{true}, and the absolute value of the residual vector $r = b - A * x$ generated by your solution. What do you observe? (If you are using Julia rather than Matlab, there is a package you can use to read the input file available at:
`https://github.com/JuliaIO/MAT.jl`)

(d) By avoiding pivoting, we were able to get a fast, $\mathcal{O}(n)$ algorithm. However, what do you observe about the accuracy of the solution and how might you explain it?

(e) If we introduce partial pivoting into our algorithm can we guarantee that the algorithm will still have $\mathcal{O}(n)$ complexity for a general instance of this problem (not just the one in part (c))? If not, what about $\mathcal{O}(n^2)$ complexity?

(f) **Ungraded, but interesting to try:** Sometimes, though not always, we may be able to pair the faster, albeit problematic, algorithm with an iterative method (a class of algorithms we will talk more about later in this course) to clean up the solution. One simple methodology is called iterative refinement. Specifically, given an approximate solution $\tilde{x}$ such that $A\tilde{x} \approx b$, we can refine the solution via the following procedure:

(a) Compute $r = b - A\tilde{x}$

(b) Solve the linear system $Az = r$

(c) Let $x = \tilde{x} + z$

If we solve $Az = r$ exactly, then x will be a solution. However, in general since that is also a problem involving A the solution for z is only approximate and it requires a more careful analysis to determine how much we may improve the solution $\tilde{x}$. In fact, it could even be repeated multiple times. If you are so inclined, try this out for this problem and see if it helps.

Question 3:

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 has all ones one the diagonal and in the last column, and all the entries below the diagonal are -1.

(a) Consider computing an LU decomposition of A (with or without partial pivoting, the answer will be the same). 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) Can you think of a different style of pivoting that would avoid the issues highlighted here? Briefly outline/describe your proposed procedure.

Note that this matrix shows off 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.