Linear Algebra and its Applications
Assignment 3

Due: 22 February 2024

Write a your own LU solver, based on Gaussian elimination with partial pivoting, in Python. The input is a square invertible matrix A and a right hand side vector b (both randomly generated). The output should be L, U and the (row) permutaion matrix and the solution to the problem $Ax = b$. Once you are sure that the code is working properly you should experiment with various sizes and note the time taken. Prepare comparison tables (one for factorization and the other for the solution, i.e., forward and backword substitution) that shows the time needed for your algorithm and also for the SciPy's LU solver. For each system you should also note down how far were you from the actual solution. [20 points]

To be precise, here are some guidelines.

- There is a built-in LU solver in SciPy. However, you should write your own without calling any readily available routine from any of the Python libraries.
- Use the NumPy random library to generate an $n \times n$ real matrix for various values of n, like $n = 10, 50, 100, 400, 800, 1000, \ldots$ etc. (take at least 5 different values, more the better)
- Implement the Gaussian elimination with partial pivoting algorithm. Do maintain an array that keeps track of pivots. In case you reach a situation where there is no (nonzero) pivot available in that column then you should exit the program, saying "unable to find nonzero pivot".
- The ideal output of the program should be the permutation matrix P, the unit lower triangular L and the upper triangular U. Of course, you don't want to print these matrices individually.
- Measure the time taken just for the factorization (don't count matrix initialization, verification etc.) and separately for substituions.
- Prepare the following tables: matrix norm of $PA - LU$ and the vector norm of $Ax_0 - b$ for both the solvers. If the norm is zero then you are sure that there were no precision errors. Compare these values for the in-built solver with yours.

Implementation guidelines

- You should write an $helper\_functions.py$" which will have all the auxiliary functions for your implementation.
- Document each of your functions well (at least one line for each).
- Write the "$results.ipynb$" file, which will import the "$helper\_functions.py$" file and display the results.
- use an understandable naming convention for your functions (don't use one alphabet for defining functions and variables).
- The pdf of the results notebook should be named "$your\_name\_roll\_no.pdf$"

Note: Your submission should be a single PDF containing your Juypter Notebook, python files and comparison tables. Make sure that unnecessary things like, error messages, trial runs etc. are not included in the submission.