Linear Algebra and its Applications
Assignment 4

Due: 19 March 2024

In this assignment you will experiment with various pivoting strategies for Gaussian elimination and Cholesky factorization. You will also compare the results of your algorithms with the ones obtained using SciPy's solvers.

Let A be the $n \times n$ matrix whose entries are

$$a_{ij} = \frac{1}{i+j-1}$$

and b be a randomly generated $n \times 1$ right hand side.

Write a your own GE solvers in Python that employ the following pivoting strategy. You should also define a tolernace ϵ such that if any nonzero entry available as a pivot is less than ϵ (you may want to experiment with values in the range of 10^{-6} to 10^{-10}) then the program will quit.

1. `GEPP`: the Gaussian algorithm with partial pivoting by row, which consists, at each step k, in determining an index $i_0 (k \leq i_0 \leq n)$ such that
$$|a_{i_0 k}| = \max_{k \leq i \leq n} |a_{ik}|.$$
2. `GERP`: the Gaussian algorithm with rook pivoting, i.e., at each step k the pivot is chosen from the kth column as well as the row.
3. `GECP`: GE with complete pivoting, i.e., the pivot is chosen from the remaining $(n-k) \times (n-k)$ submatrix.
4. `CHOP`: Cholesky factorization with pivoting, i.e., at each step the absolute largest is chosen from the diagonal.

You may solve using LU decomposition or modify b along the way and finally use back substitution. Run all these algorithms for various values of n (e.g., $n = 8, 9, 10, \ldots, 13$ etc). After each run calculate the 2-norm of the residue (i.e., $||Ax - b||$). Finally, make a comparison table of residue norm for each algorithm versus the value of n. Along with your solvers also add results from LU and Cholesky solvers of Scipy.

Can you find a reasonable bound on the condition number? Justify your answer. [30 points]

Implementation guidelines

- You should use in-built routines for matrix multiplication, norm calculation etc.
- You should write an $\mathit{helper\_functions.py}$" which will have all the auxiliary functions for your implementation.
- Document each of your functions well (at least one line for each).
- Write the "$\mathit{results.ipynb}$" file, which will import the "$\mathit{helper\_functions.py}$" 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 PDF output of your Juypter Notebook and the python files. Make sure that unnecessary things like, error messages, trial runs etc. are not included in the submission. Do upload a compressed archive. Make sure that a PDF file is uploaded.