M.Sc. in High-Performance Computing 5633A - Numerical Methods for High-performance Computing Programming Assignment 4

Kirk M. Soodhalter (`ksoodha@maths.tcd.ie`)
School of Mathematics, TCD

Rules

To submit, make a single tar-ball with all your code and a pdf of any written part you want to include. Submit this via Microsoft Teams by 8. December, 2020. Late submissions without prior arrangement or a valid explanation will result in reduced marks.

Question

For this assignment, we are going to explore Gaussian elimination and fixed-point iterations. Please implement the function

$$[\mathtt{L},\mathtt{U},\mathtt{P}] = \mathtt{mylu}(\mathtt{A}),$$

which should execute Gaussian elimination with partial pivoting as described in the lecture.

For the following, develop Matlab scripts to carry out the requested tasks.

1. Recall that even LU with partial pivoting can behave poorly for certain pathological matrices, such as ones with the structure

$$\begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}$$

with the behavior becoming more pronounced as the dimension n gets larger. For each $n = 5, 10, 20, 30, 40, 50, 60$, perform the following experiments:

- Generate matrices like the one above but replace the -1's with random uniform negative numbers in $[-1, 0]$ using `-rand()`. For each n generate 1000 such matrices, compute the pivoted LU factorization, and compute the growth factor.
- For each n, display a histogram of the orders of magnitude of the growth factors. You can use `round(log10(growthRate),1)`.
- For each n, compute the average growth rate from your experiments and plot them against n using `semilogy()`. Interpret what you see in this figure.

2. Use the included file to generate the finite difference matrix approximation of the three-dimensional Laplacian operator, `L=hw4_3d2ndDeriv(m)`. Using the other attached file, extract the tridiagonal of this matrix, `M=tridiag(L)`. Consider the fixed-point iteration induced by the splitting $L = M - N$.

   - Implement this fixed-point iteration but without computing N. *Hint*: consider the Richardson iteration version of the FPI. In your implementation, you should precompute the stable partially-pivoted LU factorization of M and use it to apply M^{-1} at each iteration.
   - What is the upper bound on the convergence rate? *Note*: use `eig(full(B))` to get the eigenvalues of a sparse matrix B.
   - For $\vec{x}_0 = \vec{0}$ and $\vec{b} = \vec{1}$, solve $L\vec{x} = \vec{b}$ using this FPI, to a relative residual tolerance of 10^{-8}. Using `semilogy()`, plot the relative residual at each iteration. How does this rate compare with the convergence rate bound?