MA 3257 – SPRING 2023 C-TERM
HOMEWORK III (DUE FEB 3RD, 2023)

Problem. (25 points). In the following descriptions, **Algorithm** and **Step** are always referring to the **Algorithm** in the notes for Lecture 9 on canvas.

Submit your .m files.

(1) (25 points) Implement the **Algorithm**. Your script may include the following checkpoints:

(a) For **Step 1**, after $i = 1$ is finished, print out what the matrix looks like. You expect the first column to have only one nonzero entry at a_{11} and the rest are zeros, i.e.

$$\left[\begin{array}{ccccc|c} a_{11} & a_{12} & . & . & a_{1n} & a_{1,n+1} \\ 0 & a_{22} & . & . & a_{2n} & a_{2,n+1} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & a_{n-1}a_{2} & . & . & a_{nn} & a_{n,n+1} \end{array}\right]$$

(b) After all row operations are done, that is, the entirety of **Step 1** in the algorithm is carried through all $i = 1, \ldots n-1$, print out what the row reduced matrix looks like. You expect it to be **upper triangular**, i.e.,

$$\left[\begin{array}{ccccc|c} a_{11} & a_{12} & . & . & a_{1n} & a_{1,n+1} \\ 0 & a_{22} & . & . & a_{2n} & a_{2,n+1} \\ . & 0 & . & . & . & . \\ . & . & 0 & . & . & . \\ 0 & . & . & 0 & a_{nn} & a_{n,n+1} \end{array}\right]$$

(2) (Extra credit: 5 points) The test cases for your script should be **square matrices** `A=rand(n,n)` and a random vector `b=rand(n,1)`. For this part, let $n = 8$. Run `A_u=triu(A)` and `A_l=tril(A)` to extract the upper and lower triangular part of A respectively.

(a) Run your solver on `A_u` and a random vector b. **Step 1** of the **Algorithm** is moot at this point (do you agree?). Print out the "before" and "after". Print out the solution.

(b) Run your solver on `A_l` with the same b as above. You should expect that after **Step 1** of the algorithm, the resulting matrix is diagonal. Print out the "before" and "after". Print out the solution.

(3) (Extra credit: 5 points) For each $n = 2, 2^2, 2^3, \ldots, 2^{10}$, generate $N = 20$ random matrices of dimension n.

(a) (2 points) Record the average runtime of your program, for each n and then plot this average against n. We expect the runtime to increase.

(b) (2 points) Record the average runtime of the MATLAB built-in solver $A\backslash b$ (A backslash b). Plot this average against n in the same figure as above.

(c) (1 points) Let x be the solution found by your solver, and x_{MATLAB} be that found by the built-in solver. For each n, compute $err_i(n) = \|x - x_{MATLAB}\|_2$ via `norm(x-x_matlab,2)`, for $i = 1, 2, \ldots, 20$; average over these to obtain

$$err(n) = \frac{1}{N}\sum_{i=1}^{N} err_i(n)$$

Plot $err(n)$ vs n.