Programming Assignment #1

Due: Mar. 15, 2024

Please show all work in order to get full credit.

Matlab Problem:

M1) Compute the inverse of *any* matrix, if one exists, using the Gauss-Jordan method. If the matrix is rectangular or non-invertible, your program must report back an error message. Your program should also take care of the case where the matrix is a single scalar.

You may **not** use any built-in function such as `rank`, `det`, `eig`, and `inv` to determine the invertibility of or to compute the inverse, of the matrix. You must implement the Gauss-Jordan method as stated in class to find inverse. You can, if needed, use built-in functions such as `find`, `isempty`, `diag`, `eye`, `abs`, `size` or other built-in functions that do not directly determine compute the inverse if necessary. If you are not clear as to which built-in functions you can or cannot use, please consult your instructor.

Finally, you certainly **cannot** download code from the internet and submit it as your own, even you make minor modifications. You must write your own code by yourself from scratch. You are **strongly encouraged** to discuss with your classmates, your instructor, and/or TA about how the code can be implemented, but again, you must write your own code.

Hints:

- Implemention of the Gaussian and Gauss-Jordan elimination procedure may require the use of nested loops, e.g.

```
while(···)
    for m = ···
        ⋮
    end
end
```

- Error messages can be displayed using the built-in function `error`. Please type `help error` for further information
- You must write this as a function, e.g.

$$\texttt{function}\,[\texttt{Ainv}] = \texttt{GJinv}(\texttt{A})$$

  where the input of the input is the matrix you want to invert, and the output is the inverse of the matrix, if the inverse exists. This makes it easier for your TA to check your code.
- Below are four matrices with different "strange" properties which you want to try your code on to check if it works properly or not before submission. The four matrices are included inside "pa01_GJinv_part_testing_matrices.mat".

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}, \quad \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ -4 & -8 & -10 \end{bmatrix}, \quad \begin{bmatrix} 1 & 2 & 3 & 3 \\ 2 & 4 & 5 & 4 \\ 3 & 5 & 6 & 5 \\ 7 & 8 & 9 & 6 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & -2 \\ 2 & 4 & -10 \\ 2 & 4 & -5 \end{bmatrix} \tag{1}$$

- Depending on your implementation, numerical instability may occur due to the finite precision of machines and causes your code unable to identity some singular matrices and produce output with very large numbers instead. A quick fix for this is to set a small number ϵ and instead of $x = 0$, set $x \leq \epsilon$ as the condition. Note that both the issue and the actual solution used in practice are much more complicated.

Grading policy:

- 10 test cases, 4 of which are from (1): 9% each, 90% in total.
- 10% for performance. Your code would be timed and compared with your classmates.

You need to upload submit your Matlab assignment (code) via the E3 system.