MTH 261 — Computational Linear Algebra (S26) Coding Homework 2

Due: By **11:59pm** on **Monday, February 16, 2026**

Name: _____

Goal. The goal of this assignment is to keep building fluency with MATLAB functions and to connect them to the linear algebra algorithms we are learning. In this homework you will compute:

- determinants of **triangular** matrices (quickly and reliably),
- inverses of **upper triangular** matrices using repeated back-substitution.

What to turn in. Upload **one** MATLAB file (`.m`) containing the two functions:

`detTri(U)` and `invUpper(U)`.

You may include a short testing section at the bottom of the same file (recommended).

Assumptions (for this assignment).

- For `detTri(U)`: U is square. Your function should *check* whether it is triangular.
- For `invUpper(U)`: U is an $n \times n$ *upper triangular* matrix. If any diagonal entry is zero, the inverse does not exist.

Restrictions. Inside your functions, do **not** use

\ `inv` `det` `rref` `linsolve`

or any built-in function that directly computes a determinant/inverse/solution. You may use them *outside* the functions to check your work.

Part 1 — Determinant of a triangular matrix: `detTri(U)`

Fact. If U is triangular (upper or lower), then

$$\det(U) = u_{11}u_{22}\cdots u_{nn}$$

(the product of the diagonal entries).

Task. Write a function `detTri(U)` that:

- returns $\det(U)$ if U is triangular (upper *or* lower),
- returns a helpful message (or throws an error) if U is not square,
- returns a helpful message (or throws an error) if U is square but not triangular.

Required checks. Verify your function on:

$$U_1 = \begin{bmatrix} 2 & 1 & -3 \\ 0 & 4 & 5 \\ 0 & 0 & -1 \end{bmatrix} \qquad \Rightarrow \qquad \det(U_1) = 2 \cdot 4 \cdot (-1) = -8,$$

and

$$L_1 = \begin{bmatrix} 3 & 0 & 0 \\ -2 & 5 & 0 \\ 1 & 7 & 2 \end{bmatrix} \qquad \Rightarrow \qquad \det(L_1) = 3 \cdot 5 \cdot 2 = 30.$$

(You may confirm with `det(U1)` outside your function.)

Part 2 — Inverse of an upper triangular matrix: `invUpper(U)`

Idea. To compute U^{-1}, solve

$$Ux^{(i)} = e_i$$

for each $i = 1, \ldots, n$, where e_i is the i-th standard basis vector. Then the solutions become the columns of the inverse:

$$U^{-1} = \begin{bmatrix} x^{(1)} & x^{(2)} & \cdots & x^{(n)} \end{bmatrix}.$$

Since U is upper triangular, each system can be solved by **back-substitution**.

Task. Write a function `invUpper(U)` that:

- returns U^{-1} if U is upper triangular with nonzero diagonal,
- handles the cases: non-square input, not upper triangular, or zero diagonal entry.

Note. You are welcome to re-use your `backsub(U,b)` logic from Coding HW1 (copy/paste into this file, or re-implement cleanly), but do **not** call MATLAB solvers.

Required check (2x2).

$$U = \begin{bmatrix} 1 & 2 \\ 0 & 2 \end{bmatrix}.$$

Compute U^{-1} by hand (quick) and verify your function produces the same result. Also check that $U \cdot U^{-1}$ is close to the identity.

Required check (3x3).

$$U = \begin{bmatrix} 2 & -1 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 5 \end{bmatrix}.$$

Verify that $\|U \cdot$ `invUpper(U)` $- I\|$ is close to 0.

Style expectations

Avoid lines of code that are hard to interpret without explanation. Use clear variable names, short steps, and brief comments.