

**MATLAB Project: LU Factorization**

Name \_\_\_\_\_

**Purpose:** To practice Lay's LU Factorization Algorithm and see how it is related to MATLAB's **lu** function.

**Prerequisite:** Section 2.5

**MATLAB functions used:** \*, **lu**; and **ludat** and **gauss** from Laydata Toolbox

**Background.** In Section 2.5, read about Lay's algorithm for calculating an LU factorization; especially study Example 2. It is imperative you understand the algorithm for calculating the matrix L before starting. In this project you will perform his algorithm on the matrices below, and see the connection between his algorithm and the one used by MATLAB's **lu** function.

$$A = \begin{bmatrix} -5 & 3 & 4 \\ 10 & -8 & -9 \\ 15 & 1 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 15 & 1 & 2 \\ 10 & -8 & -9 \\ -5 & 3 & 4 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 3 & -5 & -3 & 0 \\ 0 & -2 & 3 & 1 & -1 \\ 0 & -10 & 15 & 5 & -5 \\ 0 & 2 & -3 & -1 & 1 \\ 1 & 1 & -2 & -2 & -1 \end{bmatrix} \quad D = \begin{bmatrix} 2 & -4 & -2 & 4 \\ 6 & -9 & -3 & 7 \\ -1 & -4 & 0 & 8 \end{bmatrix}$$

$$E = \begin{bmatrix} 2 & 6 & -1 \\ -4 & -9 & -4 \\ -2 & -3 & 0 \\ 4 & 7 & 8 \end{bmatrix} \quad F = \begin{bmatrix} 1 & 3 & 0 \\ 4 & 4 & 8 \\ 1 & 2 & 3 \end{bmatrix}$$

To begin, type **ludat** to get the matrices above. For each matrix, use **gauss** to reduce the matrix to echelon form; **gauss** does only "add a multiple of one row to another" type operations -- no scaling or row exchanges. When used as shown below, **gauss** zeros out the entries directly below each successive pivot position. Type **help gauss** to learn more about how this function can be used.

1. (MATLAB) For each matrix above, use **gauss** and the algorithm in Section 2.5 to calculate an LU factorization. Record the matrix gotten from each **gauss** step; inspect those to write L; and finally, verify that LU does equal the original matrix (where U is the final matrix in your reduction).

- (a) Here is the solution for A; notice we store each intermediate matrix as U, but the final U is the one we really want:
- U = A (copy A so you can keep the original matrix and work with the copy)
  - U = **gauss(U,1)** (pivot on the first nonzero entry in row one)
  - U = **gauss(U,2)** (pivot on the first nonzero entry in row two)

The matrices produced by the commands above:                      Inspecting the matrices on the left gives L:

$$A = \begin{bmatrix} -5 & 3 & 4 \\ 10 & -8 & -9 \\ 15 & 1 & 2 \end{bmatrix} \sim \begin{bmatrix} -5 & 3 & 4 \\ 0 & -2 & -1 \\ 0 & 10 & 14 \end{bmatrix} \sim \begin{bmatrix} -5 & 3 & 4 \\ 0 & -2 & -1 \\ 0 & 0 & 9 \end{bmatrix} = U \quad L = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & -5 & 1 \end{bmatrix}$$

The following lines will store L and allow you to verify that LU does look like A:

```
L = [1 0 0;-2 1 0;-3 -5 1]
L*U, A
```

Modify the commands above for the other matrices, and record the same type of information for each:

(b)  $B = \begin{bmatrix} 15 & 1 & 2 \\ 10 & -8 & -9 \\ -5 & 3 & 4 \end{bmatrix} \sim$

- (c) After two **gauss** steps on  $C$  you will see some zero rows. This simply means all multipliers are zero after that, so put zeros in the positions that remain unfilled in  $L$ . Remember that  $L$  should have 1's on its diagonal.

$$C = \begin{bmatrix} 1 & 3 & -5 & -3 & 0 \\ 0 & -2 & 3 & 1 & -1 \\ 0 & -10 & 15 & 5 & -5 \\ 0 & 2 & -3 & -1 & 1 \\ 1 & 1 & -2 & -2 & -1 \end{bmatrix} \sim$$

$$(d) D = \begin{bmatrix} 2 & -4 & -2 & 4 \\ 6 & -9 & -3 & 7 \\ -1 & -4 & 0 & 8 \end{bmatrix} \sim$$

$$(e) E = \begin{bmatrix} 2 & 6 & -1 \\ -4 & -9 & -4 \\ -2 & -3 & 0 \\ 4 & 7 & 8 \end{bmatrix} \sim$$

$$(f) F = \begin{bmatrix} 1 & 3 & 0 \\ 4 & 4 & 8 \\ 1 & 2 & 3 \end{bmatrix} \sim$$

2. (hand) Let  $A$  be the matrix from part 1 of this project and let  $\mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ . Solve  $A\mathbf{x} = \mathbf{b}$  with the method described

in the beginning of section 2.5 of the text. The idea is that the following equations are equivalent:  $A\mathbf{x} = \mathbf{b} \iff LU\mathbf{x} = \mathbf{b} \iff L\mathbf{y} = \mathbf{b}$  and  $\mathbf{y} = U\mathbf{x}$ . Thus, if you first solve  $L\mathbf{y} = \mathbf{b}$  for  $\mathbf{y}$  and then  $U\mathbf{x} = \mathbf{y}$  for  $\mathbf{x}$ , you will have the solution to  $A\mathbf{x} = \mathbf{b}$ . Do the calculations by hand, and show work:

Step 1. Solve  $L\mathbf{y} = \mathbf{b}$  for  $\mathbf{y}$  by forward substitution:

Step 2. Using the vector  $\mathbf{y}$  from Step 1, solve  $U\mathbf{x} = \mathbf{y}$  for  $\mathbf{x}$  by back substitution:

Background on a new topic: MATLAB's `lu` function. There are two ways to use `lu`, and we will illustrate these with the matrix  $F$  above. Either way the result will not be an LU Factorization of the original matrix but rather of a different matrix  $PF$ , where  $P$  is a permutation matrix. These matters are discussed a little more in the Remarks at the bottom of page 4.

If you type `[L U P] = lu(F)`, this will not produce the factorization you got in 1(f). Instead, `lu` will first create

$$PF = \begin{bmatrix} 4 & 4 & 8 \\ 1 & 3 & 0 \\ 1 & 2 & 3 \end{bmatrix}, \text{ where } P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \text{ It will then factor } PF, \text{ obtaining } L = \begin{bmatrix} 1 & 0 & 0 \\ .25 & 1 & 0 \\ .25 & .5 & 1 \end{bmatrix} \text{ and } U = \begin{bmatrix} 4 & 4 & 8 \\ 0 & 2 & -2 \\ 0 & 0 & 2 \end{bmatrix}.$$

The product  $LU$  will equal  $PF$ , not  $F$ .

Try this for yourself: type `[L U P] = lu(F)` to see these matrices, and then type `P*F, L*U` to verify for yourself that  $PF = LU$  is true. Compare the  $L$  and  $U$  obtained here with those you got in 1(f).

If instead you type `[L1 U1] = lu(F)`,  $U_1$  will be the same  $U$  as before but the matrix  $L_1$  will be  $P^T L$ . This time  $L_1$  is not lower triangular, but the product  $L_1 U_1$  will equal the original matrix  $F$ . Try this for yourself: type `[L1 U1] = lu(F)` and observe that  $L_1$  equals  $P^T L$ , not  $L$ ; then type `L1*U1` and verify that  $L_1 U_1$  does equal  $F$ .

3. Use the **lu** function as just described with the matrices  $A$ ,  $B$ , and  $C$  discussed earlier, and record results. If necessary, recall these matrices by typing **ludat**. We do the first one as an example:

(a) For  $A = \begin{bmatrix} -5 & 3 & 4 \\ 10 & -8 & -9 \\ 15 & 1 & 2 \end{bmatrix}$ , first type **[L U P] = lu(A), L\*U, P\*A** and record results:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ .6667 & 1 & 0 \\ -.3333 & -.3846 & 1 \end{bmatrix}, U = \begin{bmatrix} 15 & 1 & 2 \\ 0 & -8.6667 & -10.3333 \\ 0 & 0 & .6923 \end{bmatrix}, P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}; LU = PA = \begin{bmatrix} 15 & 1 & 2 \\ 10 & -8 & -9 \\ -5 & 3 & 4 \end{bmatrix}$$

Then type **[L1 U1] = lu(A), L1\*U1, A** and record results:  $L_1 = \begin{bmatrix} -.3333 & -.3846 & 1 \\ .6667 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$ ,  $U_1 = U$ ,  $L_1 U_1 = A$

$$(b) B = \begin{bmatrix} 15 & 1 & 2 \\ 10 & -8 & -9 \\ -5 & 3 & 4 \end{bmatrix}$$

$$(c) C = \begin{bmatrix} 1 & 3 & -5 & -3 & 0 \\ 0 & -2 & 3 & 1 & -1 \\ 0 & -10 & 15 & 5 & -5 \\ 0 & 2 & -3 & -1 & 1 \\ 1 & 1 & -2 & -2 & -1 \end{bmatrix}$$

**Remarks.** A permutation matrix  $P$  is one obtained by rearranging the rows of an identity matrix. The effect of calculating  $PA$  is to produce that same rearrangement of the rows of  $A$ .

The algorithm used in **lu** is called "partial pivoting," and this is what causes the creation of  $PA$ . Using partial pivoting typically improves the accuracy of row operations, so all professional software for solving linear systems, like MATLAB's **lu** and backslash functions, does partial pivoting.