MAT 343 Laboratory 3
The LU decomposition

In this laboratory session we will learn how to

1. Find the LU decomposition of a matrix using elementary matrices
2. Use the MATLAB command `lu` to find the LU decomposition
3. Solve linear systems using the LU decomposition

Instructions: For your lab write-up follow the instructions of LAB 1.

Elementary Matrices and Row Reduction

An *elementary matrix* is any matrix that results from applying a single elementary row operation to the identity matrix. In this section we will discover some properties of elementary matrices and their connection to row reduction of matrices.

For example, the following three elementary matrices were obtained from the 4×4 identity matrix:

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 7 & 0 & 0 & 1 \end{bmatrix} \tag{L3.1}$$

(The row operations were, respectively: Interchange rows 1 and 3; multiply row 2 by 5; multiply row 1 by 7 and add to row 4).

We can construct each of the three types of elementary matrices using basic MATLAB commands.

Elementary Matrices of Type I:

To swap rows i and j of the $n \times n$ identity matrix, set

```
E1 = eye(n);
E1([i,j],:) = E1([j,i],:)
```

Elementary Matrices of Type II:

To multiply row i of the $n \times n$ identity matrix by the scalar c, set

```
E2 = eye(n);
E2(i,i) = c
```

Elementary Matrices of Type III:

To replace row j of the $n \times n$ identity matrix by the sum of row j and c times row i, set

```
E3 = eye(n);
E3(j,i) = c
```

Example 1

We can generate the three matrices listed in (L3.1) using the following MATLAB commands.

```
>> E1 = eye(4);
>> E1([1,3],:)=E1([3,1],:)
E1 =
     0     0     1     0
     0     1     0     0
     1     0     0     0
     0     0     0     1
>> E2 = eye(4);
>> E2(2,2)=5
E2 =
     1     0     0     0
     0     5     0     0
     0     0     1     0
     0     0     0     1
>> E3 = eye(4);
>> E3(4,1)=7
E3 =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     7     0     0     1
```

Important Property: Suppose that E is an $n \times n$ elementary matrix obtained from the identity by performing one of the elementary row operations. If A is an $n \times r$ matrix, then the matrix EA is obtained by performing the same row operation on A.

EXERCISE 1

Generate a random 4×3 matrix A (with integer entries), by typing

```
A = floor(10*rand(4,3))
```

Let `E1`, `E2` and `E3` be the elementary matrices generated in Example 1. Compute the products `E1*A`, `E2*A` and `E3*A` and compare `A` with each of the products. Describe the effect of each multiplication on the matrix A. What general pattern do you see? That is, what effect does multiplying a matrix A on the left by an elementary matrix have on the matrix A?

The LU decomposition using elementary matrices

EXERCISE 2

Enter the matrix

$$A = \begin{bmatrix} 1 & -2 & 3 \\ 2 & -6 & 5 \\ -1 & -4 & 0 \end{bmatrix}$$

(a) Determine elementary matrices **of Type III** E_1, E_2, E_3 (give names to these elementary matrices so that you can use them again later) such that $E_3E_2E_1A = U$ with U an upper triangular matrix.

(b) Compute the product $L = E_1^{-1}E_2^{-1}E_3^{-1}$. Verify that L is lower triangular with ones on the diagonal and verify that $A = LU$.

NOTE: From part (a) we have $E_3E_2E_1A = U$ and therefore $A = (E_3E_2E_1)^{-1}U = (E_1^{-1}E_2^{-1}E_3^{-1})U$. Since the elementary matrices and their inverses are lower triangular and have 1's along the diagonal, as is always true for elementary matrices of the third type, it follows that L is also lower triangular with 1's along the diagonal.

Permutation matrices

A *permutation* matrix is a square matrix that has exactly one 1 in every row and column and 0's elsewhere. In this section we will look at properties of permutation matrices.

One way to construct permutation matrices is to permute the rows (or columns) of the identity matrix. For example, we can construct

$$E = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{L3.2}$$

by using the MATLAB commands

```
p = [3,1,2,5,4];   % this is the permutation vector that defines the new order of the rows
E = eye(length(p));    % define E as the identity matrix
E = E(p,:)  % permute the rows of E according to the permutation vector p
```

The second command creates the 5×5 identity matrix, and the third command uses the vector `p` to permute its rows, so row 3 becomes row 1, row 1 becomes row 2, row 2 becomes row 3 and so on (compare these row permutations with the vector `p` defined above).

EXERCISE 3
Let E be the permutation matrix defined in (L3.2).
Generate a 5×5 matrix A with integer entries using the command

```
A = floor(10*rand(5))
```

(a) Compute the product EA and compare the answer with the matrix A. How are the two matrices related? Describe the effect on A of left multiplication by the permutation matrix E.
Compute the product AE and compare the answer with the matrix A. How are the two matrices related? Describe the effect on A of right multiplication by the permutation matrix E.

(b) Compute E^{-1} and E^T, and observe that they are also permutation matrices. What else do you observe about E^{-1} and E^T?

The *LU* decomposition in MATLAB

Not all matrices have an LU factorization because not all matrices can be reduced to upper triangular form using only row operation of Type III. However, if we allow the rows to be interchanged, then all matrices have an LU factorization (but we have to keep track of the rows we permute).

Example 2

The matrix

$$A = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 2 & 5 \\ -1 & 5 & 3 \end{bmatrix}$$

does not have an LU factorization. Zeroing out the entries below the pivot in the first column yields

$$\begin{bmatrix} 2 & 4 & 6 \\ 0 & 0 & 2 \\ 0 & 7 & 6 \end{bmatrix}$$

and it is impossible to put this matrix in triangular form without permuting the rows. However, the matrix

$$PA = \begin{bmatrix} 2 & 4 & 6 \\ -1 & 5 & 3 \\ 1 & 2 & 5 \end{bmatrix}$$

(obtained by interchanging row 2 and row 3 of A) does have an LU factorization. Here P is the permutation matrix

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

In general, for any matrix A, it is always possible to find the LU decomposition of PA, where P is an appropriate permutation matrix.

The MATLAB command

```
[L,U,P] = lu(A)
```

returns a lower triangular matrix L (with 1's on the diagonal), an upper triangular matrix U, and a permutation matrix P so that `P*A = L*U`.
For this example we have:

```
>> A=[2,4,6;1,2,5;-1,5,3]
A =
       2          4          6
       1          2          5
      -1          5          3
>> [L,U,P]=lu(A)
L =
       1          0          0
      -1/2        1          0
       1/2        0          1
U =
       2          4          6
       0          7          6
       0          0          2
P =
       1          0          0
       0          0          1
       0          1          0
```

and we can easily verify that $PA = LU$.

Remark: Note that, even when the LU factorization of A does exist, MATLAB will most likely use a permutation matrix anyway. This is because MATLAB permutes the rows so that the pivot is always the largest entry in the column. This technique, called *partial pivoting*, reduces the round off error.

Solving Systems using the LU decomposition

Given $A = LU$, we can solve $A\mathbf{x} = \mathbf{b}$ in two steps:

(a) First solve $L\mathbf{y} = \mathbf{b}$

(b) then solve $U\mathbf{x} = \mathbf{y}$.

Both systems above are triangular and therefore they can be easily solved using backward and forward substitution.
We can carry out these two steps using the "\" command in MATLAB:

```
y = L\b
x = U\y
```

However, in general, as observed before, MATLAB will return the LU factorization of PA rather than A, thus, instead of solving the system $A\mathbf{x} = \mathbf{b}$ we will solve the equivalent system

$$PA\mathbf{x} = P\mathbf{b} \tag{L3.3}$$

To solve (L3.3) using steps (a) and (b) defined above, we only need to replace $\mathbf{b}$ with $P\mathbf{b}$.

EXERCISE 4
Enter the matrix A and the vector $\mathbf{b}$ in MATLAB:

$$A = \begin{bmatrix} 2 & 2 & 4 & 6 \\ 1 & 2 & 1 & 4 \\ -4 & -3 & -7 & -8 \\ 2 & 1 & 3 & 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 6 \\ 8 \\ -11 \\ 3 \end{bmatrix}$$

The exact solution to the system $A\mathbf{x} = \mathbf{b}$ is the vector $\mathbf{x} = (-1, 8, 1, -2)^T$.

(a) Enter `[L,U,P] = lu(A)` to find the LU decomposition of the matrix PA and verify that $PA = LU$.

(b) Use the LU decomposition you found in part (a) to solve the system $A\mathbf{x} = \mathbf{b}$. Call the computed solution `x_lu`.

(c) Enter the vector `x` and compare your solution `x_lu` from part (b) with the exact solution `x` by computing `norm(x_lu - x)`
(the `norm` function gives the magnitude of the vector, that is, for a vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)^T$, the *norm* of $\mathbf{a}$ is defined as: $\text{norm}(\mathbf{a}) = \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}$).

EXERCISE 5
In this question we will compare the speed of two methods for solving the equation $A\mathbf{x} = \mathbf{b}$ when A is an invertible square matrix.
We will use the MATLAB `tic` and `toc` command to measure the computation times.

Enter:

```
A = rand(500);    x = ones(500,1);    b=A*x;
```

Important: Be sure to use semicolon after each command so that matrices and vectors are *not* displayed. Do not print or include these large matrices and vectors in your lab write-up.

(a) Solve $A\mathbf{x} = \mathbf{b}$ using the reduced row echelon form and store the solution in `x_rref`:

```
tic; R = rref([A, b]);  x_rref = R(:,end); toc
```

(b) Solve $A\mathbf{x} = \mathbf{b}$ using the LU decomposition as you did in EXERCISE 4(a)(b), and calculate the elapsed time using the `tic toc` function. Store the solution in `x_lu`. (Make sure you use semicolon; the only output should be the elapsed time). Which method is faster?

(c) Compare the solutions from parts (a) and (b) with the exact solution `x` by computing `norm(x_rref - x)` and `norm(x_lu - x)` . How accurate are the solutions from parts (a) and (b)?