Math 551 Fall 2018
Lab 6

Goal: In this assignment you will learn basic Matlab commands dealing with LU factorization, learn to find LU factorizations of matrices, and learn to find inverses and solve linear systems using the LU factorization. During the lab session, your lab instructor will teach you the necessary MATLAB commands to complete the assignment.

What you have to submit: An M-file whose contents are described in the task list below.

Reminders About the Grading Software

Remember, the labs are graded with the help of software tools. If you do not follow the instructions, you will lose points. If the instructions ask you to assign a value to a variable, you must **use the variable name given in the instructions**. If the instructions ask you to make a variable named `x1` and instead you make a variable named `x` or `X` or `X1` or any name other than `x1`, the grading software will not find your variable and you *will* lose points. Required variables are listed in the lab instructions in a gray box like this:

Variables: `x1`, `A`, `q`

At times you will also be asked to answer questions in a comment in your M-file. You must **format your text answers correctly**. Questions that you must answer are shown in gray boxes in the lab. For example, if you see the instruction

Q7: What does the Matlab command `lu` do?

your file must contain a line similar to the following somewhere

```
% Q7: It computes the LU decomposition of a matrix.
```

The important points about the formatting are

- The answer is on a single line.
- The first characters on the line is the comment character `%`.
- The answer is labeled in the form `Q`n`:`, where n stands for the question number from the gray box.

If you do not format your answers correctly, the grading software will not find them and you *will* lose points.

Tasks

Create an M-file named `m551lab06.m` in which you will complete the following tasks.

1. Start your M-file by defining the variable `A` to hold the matrix

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

   Variables: `A`

2. Add the command

```
[LA,UA]=lu(A);
```

Variables: `LA`, `UA`

Execute the M-file and examine the contents of the matrices `LA` and `UA`. Observe that `UA` is upper triangular.

Q1: Is `LA` lower triangular? Why?

Hint: Type `doc lu` and look at the difference between the commands

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

3. Now add the command

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

Variables: `L`, `U`, `P`

Execute the M-file again and look at the matrices `L`, `U` and `P`.

Q2: Is `L` lower triangular?

4. In the Matlab command window, type the commands

```
>> P'*P
>> P*P'
```

Q3: What do the results tell you about the relationship between $\mathbf{P}$ and $\mathbf{P}^T$?

5. In the Matlab command window, type

```
>> A-P'*L*U
```

and examine the result. It should be consistent with what you saw in the `lu` documentation. (Round-off errors may prevent you from seeing exactly what you expect, but it should be close.)

6. From the previous step we saw that $\mathbf{A} = \mathbf{P^T L U}$. For your own edification, you should check that this means that $\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{P}$. One way to compute the matrix on the right-hand side is to use the backslash operator: `U\(L\P)`. Add a variable named `inv_LU` to your M-file, storing the inverse of `A` computed in this way.

Variables: `inv_LU`

7. Add a command to compute the inverse by applying the `inv` function to `A`. Name the result `inv_A`.

Variables: `inv_A`

8. Define a variable called `inv_err` holding the difference between `inv_LU` and `inv_A`.

Variables: `inv_err`

Q4: Is `inv_err` the zero matrix? Why not?

9. Create a variable `b` holding the the column vector

$$\mathbf{b} = \begin{bmatrix} 14 \\ 0 \\ -8 \end{bmatrix}.$$

Variables: `b`

10. Now we're going to solve the linear system

$$\mathbf{Ax} = \mathbf{b}$$

using the LU decomposition. This can be done efficiently in Matlab by observing that

$$\mathbf{Ax} = \mathbf{b} \quad \Longleftrightarrow \quad \mathbf{P}^T\mathbf{LUx} = \mathbf{b} \quad \Longleftrightarrow \quad \mathbf{x} = \mathbf{U}^{-1}\left(\mathbf{L}^{-1}\left(\mathbf{Pb}\right)\right).$$

We'll work our way from the innermost parentheses outward and use the backslash operator instead of explicitly computing inverses. As we proceed, notice that we *never* use the expensive operations of `inv` or matrix-matrix multiplication.

Now, add the line

```
Pb = P*b;
```

beneath the definition of `b` in your M-file.

Variables: `Pb`

Run the M-file and compare `b` and `Pb` in command window.

Q5: How are the two vectors related?

11. Add two more lines to your M-file, one computing `L\Pb` and storing it in the variable `y`, and the next finishing the computation by computing `U\y` and storing it in `x`.

Variables: `x`, `y`

Run your M-file, and examine the variable `x`. If you have completed all steps correctly, `x` should be the solution to the linear system.

```
>> x

x =

    1.0000
    2.0000
    3.0000
```

If the above isn't what you see, go back and check your work.

12. Now define a new variable `z` in your M-file and assign it the value `A\b` and define `xz_err` to be the difference between `x` from the previous steps and `z`.

Variables: `z`, `xz_err`

In the command window, examine the value of `xz_err`. This time you should get *exactly* the zero vector. The reason is that the steps you took in finding `x` are exactly the same steps that Matlab took in finding `z`.