Midterm Project, due October 19

This project requires extensive programming. Start early and follow the steps outlined below. After each stage, test your program on some simple examples before moving to the next stage.

1. Write a Matlab forward substitution function *[x] = forwardSub(L, b)* and a backward substitution function *[x] = backwardSub(U, b)* for solving $Lx = b$ and $Ux = b$, respectively.

In the inputs, L is a lower triangular matrix, U is an upper triangular matrix, b is the right hand side column vector. The output is x, solution of the system.

Generate the lower and upper triangular matrix by "$A = rand(n, n);\ L = tril(A, 0);\ U = triu(A, 0);$" Compare your answers with matlab build-in linear system solver "$x = L\backslash b$ or $x = U\backslash b$", they should be the same.

2. Write a Matlab function using the Gaussian elimination without pivoting to compute the LU factorization of a matrix A, *[L, U] = computeLU(A)*. The input A is a square matrix, and the outputs are the lower triangular matrix L and the upper triangular matrix U in the LU factorization of A.

You will need the following piece of code to output L and U at the very end of your program,
"$L = eye(n) + tril(A, -1); U = triu(A, 0)$".

Check your answer by computing $L * U - A$, which should be a zero matrix.

3. Write another Matlab function *[L, U, p] = computeLUpivot(A)* which implements the Gaussian elimination with pivoting. The output should include a vector, p, recording the pivoting history in the Gaussian elimination.

For the pivoting, to find the largest value and its position in a vector, you can use the Matlab built-in function *max*. "$[maxval, colind] = max(A(k : n,\ k));\ colind = colind + k - 1;$"

To swap rows i and j of a matrix A, consider the following:

```
temp1(:)=A(i,:);
A(i,:)=A(j,:);
A(j,:)=temp1(:);
```

For the vector p, initialize it to be

```
for i=1:n
    p(i)=i;
end
```

and then pivot it at each step, by swapping two elements

```
temp2=p(i);
p(i)=p(j);
p(j)=temp2;
```

Check your answers with Matlab build-in function "$[L, U, p] = lu(A)$";

4. Test solving system of linear equations $Ax = b$, by using the LU factorization functions in Questions **2** and **3**, respectively. The forward and backward substitution functions in Questions **1** and **2** should also be used in your code. When pivoting is applied in the LU factorization, recorded in the vector p, you need first apply the same pivoting steps to the right hand side vector b before the forward and backward substitution solves.

To solve the equations with LU factorization, consider the following script:

```
[L,U]=computeLU(A);
y=forwardSub(L, b);
```

```
x=backwardSub(U, y);
```

Similar script is needed to solve with LU pivoting. To apply the pivoting steps to b, consider the following:

```
for i=1:n
    pb(i)=b(p(i));
end
```

5. Let $A = rand(n)$ and $b = A * ones(n, 1)$, so the true solution of $Ax = b$ is $x = ones(n, 1)$.

Solve $Ax = b$ by using the LU factorization without pivoting (Question **2**), by the LU factorization with pivoting (Question **3**), and by the Matlab built-in linear system solver $x = A\backslash b$, respectively. In each case, evaluate the relative difference between the computed solution and the true solution (in vector 2-norm). To get a better representation of the error, for each algorithm, do five random runs, i.e, on five random matrices A, and compute an average of the five relative errors.

Take $n = 10, 20, 30, \cdots, 200$. For each n, find the relative error of the solution as mentioned above for each algorithm. Plot the relative errors with respect to n for the three algorithms respectively in the same graph. Use *semilogy* in the plot.

You should observe in the graph that the LU factorization with pivoting and the Matlab "\" function generate less error in the solution than the LU factorization without pivoting.