HW-4

1. The following Program computes the L and U decomposition of a matrix i.e A=LU using a row-wise access of the data. If we interchange the loops i with j, we get the well-known column-wise access LU decomposition, also known as the kji form.

   i. Write the kji program. The LU produced should be the same.

   ii. Test both programs using n=500, 1000 and 2000. How does the time grow for each program; linear, quadratic, cubic?

   iii. Both programs perform identical number of operations-- so in theory the time taken to execute by each program should be the same. Tic/toc (see help tic in Matlab) measures the elapsed time of execution. If one program is faster than the other what could be the reason? Provide an explanation.

```
n=input('n=')
a=ones(n,n)+n*eye(n,n);
A=a;
b=sum(a,2);
l=eye(n);
tstart=tic
for k=1:n-1
    for i=k+1:n
        l(i,k)=a(i,k)/a(k,k);
        for j=k:n
            a(i,j)=a(i,j)-l(i,k)*a(k,j);
        end
    end
end
telapsed=toc(tstart)
U=a; L=l;
```

2. Consider the column-wise form and add partial pivoting. The program should produce 3 matrices P a permutation matrix , L and U so that PA=LU. In matlab the program that does this can be called by—[L U P]=lu(A). So your results should be identical to the Matlab program. Notice that P*P'=eye(n)—In other words the inverse of a permutation matrix is its transpose. This implies that A=P'LU. Your program should not perform explicit row interchanges; rather you should use a permutation vector P that keeps track of the interchanges and use this to perform the appropriate computations. You are allowed to use Matlab function max- [Y,I] = MAX(X); to identify the maximum in each column as well its position—to simplify your program. Use the following matrix to do your experiments:

```
for i=1:n
     for j=1:n
       a(i,j)=1/(i+j);
     end
 end
```

i. First verify that your program gives the same results for n=4 as [LL UU PP]=lu(A).

ii. Consider `b=sum(a,2)` so that the expected solution is **ones(n,1).** To solve the system Ax=b, you could solve PAx=Pb, ==➔LUx=Pb=➔ Ux=z, Lz=Pb. Write a forward and backward substitution programs and use it to solve the system for n=5,10, 20,40. Do you get **ones(n,1)** as a solution? If not explain**. You can use matlab functions to provide an explanation of the behavior that you see.**

3. We want to solve Ax=b. Solve the system by using Gaussian elimination with partial pivoting for the following linear systems:

i. $$\begin{bmatrix} 0 & 1 & -1 \\ -1 & 2 & 0 \\ -2 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

Find PLU and solve $PLUx = b$

ii. What is PLU of the matrix A. Show all steps and then verify your result by using matlab solver $[L\ U\ P] = lu(A)$

iii. What is the complexity of Gaussian elimination when applied to general nxn matrix. Show all steps in the derivation of the complexity.

iv. What is the det(A)? which method is the best in finding the determinant(EXPLAIN)

v. Find the inverse of A by using Gauss-Jordan method—eliminating both above and below the diagonal at the same time using the extended system:

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -2 \\ -2 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Verify your result by using $inv(A)$ in Matlab. Then derive the computational complexity of inverting a general nxn matrix using the Gauss-Jordan method.

vi. We want to solve Ax=b by using two different methods- $PLUx = b$ or $x = A^{-1}b$. Which one of these two methods is better? Explain why.

4. The kij form of Gaussian Elimination without partial pivoting is given by :

```
function [L U a] = kij( a )
m=size(a);
n=m(1);
for k=1:n
    for i=k+1:n
        a(i,k)=a(i,k)/a(k,k);
        for j=k+1:n
            a(i,j)=a(i,j)-a(i,k)*a(k,j);
        end
    end
end
L = tril(a,-1);
U=a-L;
L=L+eye(n);
end
```

i. Explain each step of the above program by using the matrix $\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -2 \\ -2 & 1 & 1 \end{bmatrix}$. Help tril in matlab will explain what the function does—it extracts the lower triangular part of the matrix.

ii. Modify this program to perform the kji form of LU decomposition.

iii. Write a separate function that takes as input $L\ U$, b and outputs the solution x of Ax=b. Your function should perform two steps $LUx = b ==> Lz = b, Ux = z$, i.e a forward substitution first and a backward substation next.

iv. Test your program with the system in question 1 and present the results.

5. In class we wrote the more general form of Gaussian Elimination with partial pivoting program. Is it correct? Verify by using the matrix in the program and performing all steps manually and then multiplying A= P*L*U. Show all steps.

```
%a=[0 1 2;2 0 1;1 4 1];
%n=input('n=');
%for i=1:n
 %%         a(i,j)=1/(i+j);
 %   end
%end
a=[2 1 -1 -2;4 4 1 3;-6 -1 10 10;-2 1 8 4]

A=a;
n=length(a);
piv=1:n;
l=eye(n);
P=eye(n);
permutations=0;
for k=1:n-1
  [maxv,r] = max(abs(a(k:n,k)));
   q = r+k-1;
   piv([k q]) = piv([q k]);
   a([k q],:) = a([q k],:);
  if q~=k
      permutations=permutations+1;
  end
   disp([k q r])
   disp(a)
   pause
 for i=k+1:n
     l(i,k)=a(i,k)/a(k,k);
 end

  for j=k:n
      for i=k+1:n
          a(i,j)=a(i,j)-l(i,k)*a(k,j);
      end
  end
  disp(a);
  disp(l)
end
L=l;U=a;
P=P(:,piv);
if mod(permutations,2)==0
    sign=1;
else
    sign=-1;
end
dett=1;
for i=1:n
dett=dett*a(i,i);
end
dett=sign*dett;
```