1. Use Gaussian elimination to solve by hand the system of equations

\[
\begin{bmatrix}
1 & 1 & 2 & 0 \\
3 & 4 & 6 & 1 \\
1 & -1 & 3 & -4 \\
0 & -1 & -2 & 4 \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
\end{bmatrix}
= 
\begin{bmatrix}
-1 \\
-6 \\
9 \\
-7 \\
\end{bmatrix}.
\]

Was a permutation of equations (i.e. rows) necessary? Verify that the same results are obtained at each step by demo_alg061.m.

Write

\[
\begin{bmatrix}
1 & 1 & 2 & 0 & -1 \\
3 & 4 & 6 & 1 & -6 \\
1 & -1 & 3 & -4 & 9 \\
0 & -1 & -2 & 4 & -7 \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 1 & 2 & 0 & -1 \\
0 & 1 & 0 & 1 & -3 \\
0 & -2 & 1 & -4 & 10 \\
0 & -1 & -2 & 4 & -7 \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 1 & 2 & 0 & -1 \\
0 & 1 & 0 & 1 & -3 \\
0 & 0 & 1 & -2 & 4 \\
0 & 0 & -2 & 5 & -10 \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 1 & 2 & 0 & -1 \\
0 & 1 & 0 & 1 & -3 \\
0 & 0 & 1 & -2 & 4 \\
0 & 0 & 0 & 1 & -2 \\
\end{bmatrix}.
\]

No row swaps. By backwards substitution, we obtain \( x_4 = -2 \); \( x_3 - 2x_4 = 4 \) or \( x_3 = 0 \); \( x_2 + x_4 = -3 \) or \( x_2 = -1 \); and \( x_1 + x_2 + x_3 = -1 \) or \( x_1 = 0 \). The code below confirms this.

```plaintext
A=[ 1 1 2 0 -1;
3 4 6 1 -6;
1 -1 3 -4 9;
0 -1 -2 4 -7];
demo_alg061
```
2. Repeat Problem #1 for the system of equations
\[
\begin{pmatrix}
2 & 1 & 2 & 1 \\
-2 & -1 & -5 & 1 \\
4 & 2 & -1 & 1 \\
1 & 6 & -1 & -1
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4
\end{pmatrix}
=
\begin{pmatrix}
-4 \\
12 \\
1 \\
6
\end{pmatrix}.
\]

Write
\[
\begin{pmatrix}
2 & 1 & 2 & 1 \\
-2 & -1 & -5 & 1 \\
4 & 2 & -1 & 1 \\
1 & 6 & -1 & -1
\end{pmatrix}
\rightarrow
\begin{pmatrix}
2 & 1 & 2 & 1 \\
0 & 0 & -3 & 2 \\
0 & 0 & -5 & -1 \\
0 & 0 & 11/2 & -2
\end{pmatrix}
\rightarrow
\begin{pmatrix}
2 & 1 & 2 & 1 \\
0 & 11/2 & -2 & -3/2 \\
0 & 0 & -3 & 2 \\
0 & 0 & 0 & 11/2
\end{pmatrix}
\rightarrow
\begin{pmatrix}
2 & 1 & 2 & 1 \\
0 & 11/2 & -2 & -3/2 \\
0 & 0 & -5 & -1 \\
0 & 0 & 0 & 13/5
\end{pmatrix}.
\]

One row swap. By backwards substitution, we obtain \(13x_4/3 = 13/3\) or \(x_4 = 1\); \(-5x_3 - x_4 = 9\) or \(x_3 = -2\); \(11x_2/2 - 2x_3 - 3x_4/2 = 8\) or \(x_2 = 1\); and \(2x_1 + x_2 + 2x_3 + x_4 = -4\) or \(x_1 = -1\). The code below confirms this.

```matlab
A=[ 2 1 2 1 -4; \\
-2 -1 -5 1 12; \\
4 2 -1 1 1; \\
1 6 -1 -1 6];
demo_alg061
```
3. (a) Once the $LU$-decomposition $A = LU$ of a matrix is known, the linear equation $Ax = b$ can be solved in two steps, by first using forward substitution to solve

$$Ly = b$$

and then using backward substitution to solve

$$Ux = y.$$ 

Write a Matlab script `forwback.m` which implements this solution, assuming that the matrices $L, U$ and the input vector $b$ are known.

*Hint:* The back-substitution is identical to that performed in Steps 8 & 9 of `ALG061.m`, using $U$ rather than $A$. The forward-substitution can be performed very similarly.

(b) Count the number of multiplication/divisions and addition/subtractions in your algorithm. Note that we have assumed in our $LU$-decomposition that $\ell_{ii} = 1$ for all $i$, but there are other versions of that decomposition that make other choices, e.g. $u_{ii} = 1$ for all $i$. Rewrite your Matlab script `forwback.m` to work for a general $LU$-decomposition, by making an additional $n$ divisions.

(a) To forward substitute for $y$, for each of $i = 1, \ldots, n$ in order take $y_i := \ell_{ii}^{-1}(b_i - \sum_{j=1}^{i-1} \ell_{ij}y_j)$. Backward substitution for $x$ follows exactly as in Gaussian elimination.

(b) In forward substitution, for each $i = 1, \ldots, n-1$ there are $i$ multiplications, and for each $i = 1, \ldots, n$ there are $i-1$ additions. In other words there are $\sum_{i=1}^{n-1} i = n(n-1)/2$ multiplications and $\sum_{i=1}^{n} (i-1) = n(n-1)/2$ additions. In backward substitution we have something similar: $\sum_{i=1}^{n} i = n(n+1)/2$ and $\sum_{i=1}^{n} (i-1) = n(n-1)/2$ multiplications and additions respectively. Thus we total $n^2$ multiplications and $n^2 - n$ additions.

Possible implementation of `forwback.m`:

```matlab
% forward substitution
Y=b;
% next step unnecessary if L(1,1)=1
Y(1)=Y(1)/L(1,1);
Y
for I=2:N
    SUM= 0;
    for J=1:I-1
        SUM = SUM - L(I,J) * Y(J);
    end
    % division in next step unnecessary if L(I,I)=1
    Y(I)= (Y(I)+ SUM )/ L(I,I);
end
Y

% backward substitution
X=Y;
X(N) = Y(N) / U(N,N);
X
for K = 1:NN
    I = NN-K+1;
    SUM = 0;
    for KK = I+1:N
        SUM = SUM - U(I,KK) * X(KK);
    end;
    X(I) = (X(I)+SUM) / U(I,I);
end;
```
4. Consider the linear problems $\mathbf{Ax} = \mathbf{b}_k$, $k = 1, 2, 3$ for the $8 \times 8$ matrix

$$
\mathbf{A} = \begin{pmatrix}
-2 & 0 & -1 & 1 & 1 & 3 & 1 & -2 \\
-6 & -3 & -3 & 3 & 3 & 9 & 4 & -5 \\
4 & 3 & 4 & -3 & -2 & -6 & -1 & 3 \\
0 & 3 & 4 & -1 & 0 & -1 & -1 & 0 \\
-4 & 3 & 0 & 1 & 1 & 5 & 3 & -6 \\
0 & 6 & 4 & -2 & -1 & -2 & 3 & -5 \\
-6 & 0 & -1 & 2 & 1 & 6 & 7 & -9 \\
-8 & -9 & -6 & 6 & 5 & 10 & 1 & -7
\end{pmatrix}
$$

and

$$
\mathbf{b}_1 = (-4, -6, 3, -9, -11, -4, -3, 0)^T \\
\mathbf{b}_2 = (-1, -5, 3, 7, 0, 1, -6, -3)^T \\
\mathbf{b}_3 = (-3, -4, 5, 0, -9, -5, -8, -6)^T
$$

(a) Solve the first problem $\mathbf{Ax} = \mathbf{b}_1$ and also find an $LU$-decomposition of $\mathbf{A}$, by using demo_alg061.m, 
(b) Solve the second two problems $\mathbf{Ax} = \mathbf{b}_k$, $k = 2, 3$ by using the $LU$-decomposition from part (a) and your algorithm forwback.m from Problem #3.

(a) Code:

```
AA = [-2  0 -1  1  1  3  1 -2; 
  -6 -3 -3  3  3  9  4 -5; 
   4  3  4 -3 -2 -6 -1  3; 
   0  3  4 -1  0 -1 -1  0; 
  -4  3  0  1  1  5  3 -6; 
   0  6  4 -2 -1 -2  3 -5; 
  -6  0 -1  2  1  6  7 -9; 
  -8 -9 -6  6  5 10  1 -7];
XX = [2; -2; -1; -1; 0; -1; 1; -1];
bb = [-4; -6; 3; -9; -11; -4; -3; 0];
A = [AA, bb]
demo_alg061
```

Output:

```
L =
  1   0   0   0   0   0   0   0
  3   1   0   0   0   0   0   0
-2 -1   1   0   0   0   0   0
  0 -1   2   1   0   0   0   0
  2 -1   1   0   1   0   0   0
  0 -2   2   0   1   1   0   0
  3   0   1   0   2   1   1   0
  4   3 -1   1 -1   2 -2   1

U =
-2   0 -1   1   1   3   1 -2
  0 -3   0   0   0   1   1
  0   0   0 -1   0   2   0
  0   0   0  1 -1  -4   1
  0   0   0   0 -1 -1   0 -1
  0   0   0   0 -1   1   1 -2
  0   0   0   0   0   1   1
  0   0   0   0   0   0   0   2

X =
  2
```
(b) Code:

```matlab
bb = [-1; -5; 3; 7; 0; 1; -6; -3]
forwback
```

Output:
```
Y =
  -1
  -2
  -1
   7
   1
  -2
  -2
   0

X =
  -2
   0
   1
  -1
  -2
   0
```

Code:

```matlab
bb = [-1; -5; 3; 7; 0; 1; -6; -3]
forwback
```

Output:
```
Y =
  -1
   4
  -3
  -2
  -1
   2
   2

X =
   1
  -1
   1
   1
   0
   1
   1
```
5. (a) Write a Matlab script `gaussjord.m` to solve the linear system $Ax = b$ by the Gauss-Jordan method, as described either in the classnotes or in Homework 6.1.14 of Burden & Faires, 10th Ed.

*Hint:* Only a few minor alterations of `demo_alg061.m` are required. Note that in Matlab one can loop over $J$ between 1 and $N$ with $J \neq I$ by the following:

```matlab
for J=[1:I-1,I+1:N]
    ...
end
```

(b) Use this algorithm to solve the linear equations in Problems # 1 & 2.

(a) Possible implementation of `gaussjord.m`:

```matlab
% GAUSS-JORDAN ELIMINATION WITH BACKWARD SUBSTITUTION ALGORITHM
% To solve the n by n linear system for augmented matrix A

single(A)
pause
N=min(size(A));
L=zeros(N,N);
TRUE = 1;
FALSE = 0;
OK=TRUE;
% STEP 1
% Elimination Process
NN = N-1;
M = N+1;
ICHG = 0;
I = 1;
while OK == TRUE & I <= N
    % STEP 2
    % use IP instead of I
    IP = I;
    while IP <= N & abs(A(IP,I)) <= 1.0e-20
        IP = IP+1;
    end;
    if IP == M
        OK = FALSE;
    else
        % STEP 3
        if IP ~= I
            for JJ = 1:M
                C = A(I,JJ);
                A(I,JJ) = A(IP,JJ);
                A(IP,JJ) = C;
            end;
            ICHG = ICHG +1;
            ['interchange rows ', num2str (I), ' and ', num2str (IP)]
            single(A)
pause
        end;
    end;
    % STEP 4
    for J = [1:I-1,I+1:N]
        % STEP 5
        % use XM in place of m(J,I)
        XM = A(J,I)/A(I,I);
        % STEP 6
        %for K = [1:I-1,I+1:M]
        for K = I+1:M
            A(J,K) = A(J,K) - XM * A(I,K);
        end;
        % Multiplier XM could be saved in A(J,I).
        A(J,I) = 0;
        %A(J,I)= XM;
    end;
end;
```

I = I+1;
single(A)
pause
end;
if OK == TRUE
  % STEP 7
  if abs(A(N,N)) <= 1.0e-20
    OK = FALSE;
  else
    % STEP 8
    % solve the diagonal system
    X=A(:,M)
pause
    for I=N:-1:1
      X(I) = X(I) / A(I,I);
      X
      pause
    end
  end
  OUP = 1;
  fprintf (OUP , '

with %d row interchange(s)
', ICHG);
end;
if OK == FALSE
  fprintf (1,' System has no unique solution 
 ');
end;

(b) Code:

A = [ 1 1 0 -2 3;
-2 0 -1 3 -7;
1 3 0 -7 11;
2 2 -2 6 -16]
gaussjord

We find x = (0, -1, 1, -2).

A = [-1 2 2 4 -20;
-2 3 2 -2 3;
1 -1 0 -1 5;
0 -5 -3 3 -11]
gaussjord

We find x = (2, 1, -2, -4).
6. Show that the Gauss-Jordan method requires

\[ \frac{n^3}{2} + n^2 - \frac{n}{2} \] multiplications/divisions

and

\[ \frac{n^3}{2} - \frac{n}{2} \] additions/subtractions.

In STEP 5 and STEP 6 we count \( n(n-1) + (n-1) \sum_{i=1}^{n}(n+1-i) = (n-1) \sum_{i=1}^{n}(i+1) = \frac{n^3}{2} + n^2 - 3n/2 \) multiplications and \( (n-1) \sum_{i=1}^{n}(n+1-i) = (n-1) \sum_{i=1}^{n}i = \frac{n^3}{2} - n/2 \) additions; in STEP 8 we count \( n \) multiplications. Thus the total required number of multiplications and additions are as given.
7. Consider the Frobenius matrix

\[
F_k = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & \ddots & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & -\ell_{k+1,k} & \ddots & 0 \\
0 & 0 & 0 & 0 & \ddots & 0 \\
0 & 0 & 0 & -\ell_{n,k} & 0 & 0 & 1
\end{pmatrix}.
\]

(a) Verify by direct matrix multiplication that

\[
F_k^{-1} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & \ddots & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & \ell_{k+1,k} & \ddots & 0 \\
0 & 0 & 0 & 0 & \ddots & 0 \\
0 & 0 & 0 & \ell_{n,k} & 0 & 0 & 1
\end{pmatrix}.
\]

(b) Prove by induction on \(k\) for \(1 \leq k \leq n\) that

\[
F_1^{-1}F_2^{-1} \cdots F_k^{-1} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
\ell_{2,1} & 1 & 0 & 0 & 0 & 0 \\
\vdots & \ell_{3,2} & \ddots & 0 & 0 & 0 \\
\vdots & \vdots & \ddots & 1 & 0 & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots & 0 \\
\ell_{n,1} & \ell_{n,2} & \cdots & \ell_{n,k} & 0 & 0 & 1
\end{pmatrix}
\]

and thus

\[
L = F_1^{-1}F_2^{-1} \cdots F_{n-1}^{-1} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
\ell_{21} & 1 & 0 & 0 & 0 \\
\vdots & \ell_{32} & 1 & 0 & 0 \\
\vdots & \vdots & \ddots & \ddots & 0 \\
\ell_{n1} & \ell_{n2} & \cdots & \ell_{n,n-1} & 1
\end{pmatrix}.
\]

(a) Observe that \((F_k^{-1}F_k)_{ij} = \sum_{l=1}^{n} (F_k^{-1})_{il} (F_k)_{lj} = (F_k)_{ii} (F_k)_{ij} - (F_k)_{ik} (F_k)_{kj} = (\text{id}_n)_{ij}\).

(b) Note that \(F_k^{-1} : e_k \mapsto e_k + \sum_{i=k+1}^{n} \ell_{i,k} e_i\) and \(F_k^{-1} : e_j \mapsto e_j\) for \(j \neq k\). Consider the action of \(F_{n-2}\) on the columns of \(F_{n-1}^{-1}\). Thus the first \(n-3\) columns are \(e_1, \ldots, e_{n-3}\) and don’t change. The \((n-2)\)nd column is \(F_{n-2}^{-1}e_{n-2} = e_{n-2} + \ell_{n-1,n-1}e_{n-1} + \ell_{n,n-1}e_n\). The \((n-1)\)st column is \(F_{n-2}^{-1}(e_{n-1} + \ell_{n,n-1}e_n) = \ell_{n-1,n-1}e_{n-1} + \ell_{n,n-1}e_n\). And the \(n\)th column is \(F_{n-2}^{-1}e_n = e_n\). This proves the base case. Suppose our representation holds for \(F_{k-1}^{-1} \cdots F_{n-1}^{-1}\), and we consider the action of \(F_{k-1}^{-1}\) on its columns. The first \(k-2\) columns are \(e_1, \ldots, e_{k-2}\) and don’t change. The \((k-1)\)st column is \(e_{k-1} + \sum_{i=k}^{n} \ell_{i,k} e_i\). For \(k \leq j \leq n\) the \(j\)th column is \(F_{k-1}^{-1}(e_j + \sum_{i=j+1}^{n} \ell_{i,j} e_i) = e_j + \sum_{i=j+1}^{n} \ell_{i,j} e_i\), or in other words, invariant. This proves the representation for \(F_{k-1}^{-1} \cdots F_{n-1}^{-1}\) and we’re done.
8. If Gaussian elimination can be carried out successfully on the first $k-1$ rows of a square matrix $A$ without pivoting, then show that the pivot element at the $k$th step is given by 

$$u_{kk} = \frac{\det A_k}{\det A_{k-1}},$$

where $A_k$ is the $k$th leading principal submatrix of $A$. Use this result to prove that Gaussian elimination can be carried for square matrix $A$ without row pivoting if and only if the leading principal submatrices $A_k$ are non-singular for all $k = 1, ..., n-1$.

*Hint*: Explain why $\det(A^{(j)}_k) = \det A_k$ for all $k = 1, ..., n$ and for all $j = 1, ..., n^* - 1$, where $A^{(j)}$ is the sequence of matrices obtained in Gaussian elimination and $n^*$ is the first value of $j$ for which $u_{jj} = 0$ or $n^* = n$ if $u_{jj} \neq 0$ for all $j = 1, ..., n-1$.

The sequence of matrices $A^{(j)}$ in Gaussian elimination are obtained by elementary row operations, subtracting from one row a multiple of another row, which preserve determinants. Thus, $\det(A^{(j)}) = \det(A)$ as long as the $j$th step is reached. The same is also true for the $k$th leading principal submatrices $A^{(j)}_k$, so that $\det(A^{(j)}_k) = \det(A_k)$ as long as the $j$th step is reached.

At the beginning of the $k$th step of Gaussian elimination, we know $A^{(k)}_k$ is upper triangular and that $\det(A^{(k)}_k) = \prod_{i=1}^{k} a^{(i)}_{ii} = \prod_{i=1}^{k} u_{ii}$ while also $\det(A^{(k)}_{k-1}) = \prod_{i=1}^{k-1} a^{(i)}_{ii} = \prod_{i=1}^{k-1} u_{ii}$ for $k > 1$. Thus we immediately get $u_{kk} = \frac{\det(A^{(k)}_k)}{\det(A^{(k)}_{k-1})} = \frac{\det(A_k)}{\det(A_{k-1})}$.

One can complete the $k$th step and obtain $A^{(k+1)}$ if and only if $u_{kk} \neq 0$ which, by the previous result, is equivalent to $\det(A_k) \neq 0$. Thus, Gaussian elimination will proceed successfully to $A^{(n^*)}$ where $n^*$ is the first value of $k$ for which $u_{kk} = 0$ or $n^* = n$ if $u_{kk} \neq 0$ for all $k = 1, ..., n-1$. Thus, Gaussian elimination will succeed to obtain $A^{(n)} = U$ if and only if $\det(A_k) \neq 0$ for all $k = 1, ..., n-1$. 

10