Homework 2

Some problems are adapted from Strang I.3 and I.4. The original problem numbers are provided.

Linear Systems and LU Factorization

1. (adapted from I.3 5) Show possible number of solutions for each linear systems $A_i\mathbf{x} = \mathbf{b}$, whose left hand side A_i is given as following:

$$A_1 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 1 & 2 & 6 \\ 2 & 4 & 7 \\ 3 & 6 & 5 \end{bmatrix} \qquad A_3 = \begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 5 & 7 \end{bmatrix} \qquad A_4 = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 6 & 7 \end{bmatrix}$$

A_1 is invertible, and has unique solution. A_2's rank is less than column and row dimensions. It has 0 or infinite number of solutions. A_3's rank is equal to the column dimension. It has 0 or 1 solution. A_4's rank is equal to the row dimension. It has infinite number of solutions.

2. For each of the following statements, indicate whether the statement is true or false.

   $\boxed{\textbf{T}}$/**F** If a triangular matrix has a zero on its main diagonal, then it is necessarily singular.

   $\boxed{\textbf{T}}$/**F** The product of two upper triangular matrices is upper triangular.

   $\boxed{\textbf{T}}$/**F** Once the LU factorization of an $n \times n$ matrix has been computed to solve a linear system, then subsequent linear systems with the same matrix but different right-hand-side vectors can be solved in $O(n^2)$ time without refactoring the matrix.

3. (adapted from I.4 3) What lower triangular matrix E puts A into upper triangular form $EA = U$? Multiply by $E^{-1} = L$ to factor A into LU:

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

$$E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$L = E^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 3 & 1 \end{bmatrix}$$

$$A = LU$$

4. (adapted from I.4 4) LU factorization is generally derived using elimination matrices such as E_1 and E_2 below, which consist of the identity matrix plus the negatives of sub-diagonal elements of one column of L. These matrices have

the nice property that their inverses are trivial to compute, e.g., $E_1^{-1} = \begin{bmatrix} 1 & & \\ a & 1 & \\ b & 0 & 1 \end{bmatrix}$, and $E_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & c & 1 \end{bmatrix}$ This problem shows how the elimination matrix inverses multiply to give L. You see this best when $A = L$ is already lower triangular with 1's on the diagonal. Then $U = I$:

Multiply $A = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & c & 1 \end{bmatrix}$ by $E_1 = \begin{bmatrix} 1 & & \\ -a & 1 & \\ -b & 0 & 1 \end{bmatrix}$ and then $E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -c & 1 \end{bmatrix}$.

(a) Multiply $E_2 E_1$ to find the single matrix E that produces $EA = I$.

(b) Multiply $E_1^{-1} E_2^{-1}$ to find the matrix $A = L$.

The multipliers a, b, c are mixed up in $E = L^{-1}$ but they are perfect in L.

$$E_2 E_1 = \begin{bmatrix} 1 & 0 & 0 \\ -a & 1 & 0 \\ ac - b & -c & 1 \end{bmatrix}$$
$$E_1^{-1} E_2^{-1} = L = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & c & 1 \end{bmatrix}$$

5. (adapted from I.4 8) Tridiagonal matrices have zero entries except on the main diagonal and the two adjacent diagonals. Factor these into $A = LU$. Symmetry further produces $A = LDL^T$, where D is a diagonal matrix.

$$A_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix} \qquad A_2 = \begin{bmatrix} a & a & 0 \\ a & a+b & b \\ 0 & b & b+c \end{bmatrix}$$

$$A_1 = LU = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & -2 \end{bmatrix} = LDL^T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & -2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$
$$A_2 = LU = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} a & a & 0 \\ 0 & b & b \\ 0 & 0 & c \end{bmatrix} = LDL^T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

6. (adapted from I.4 11) In some data science applications, the first pivot is the *largest number* $|a_{ij}|$ in A. Then row i becomes the first pivot row $\mathbf{u}_1^*$. Column j is the first pivot column. Divide that column by a_{ij} so ℓ_1 has 1 in row i. Then remove that $\ell_1 \mathbf{u}_1^*$ from A.
This example finds $a_{22} = 4$ as the first pivot ($i = j = 2$). Dividing by 4 gives ℓ_1:

$$\begin{bmatrix} 1 & 2 \\ 3 & \mathbf{4} \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1 \end{bmatrix} \begin{bmatrix} 3 & \mathbf{4} \end{bmatrix} + \begin{bmatrix} -1/2 & 0 \\ 0 & 0 \end{bmatrix} = \ell_1 \mathbf{u}_1^* + \ell_2 \mathbf{u}_2^* = \begin{bmatrix} 1/2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 3 & 4 \\ -1/2 & 0 \end{bmatrix}$$

For this A, both L and U involve permutations. P_1 exchanges the rows to give L. P_2 exchanges the columns to give an upper triangular U. Then $P_1 A P_2 = LU$.
Permuted in advance $P_1 A P_2 = \begin{bmatrix} 1 & 0 \\ 1/2 & 1 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 0 & -1/2 \end{bmatrix} = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$.
Question for $A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$: Apply complete pivoting to produce $P_1 A P_2 = LU$.

Perform the (both column and row) permutations as shown, $P_1 A P_2 = \begin{bmatrix} 4 & 2 \\ 3 & 1 \end{bmatrix}$. Then perform the LU decomposition:
$P_1 A P_2 = \begin{bmatrix} 1 & 0 \\ 3/4 & 1 \end{bmatrix} \begin{bmatrix} 4 & 2 \\ 0 & -1/2 \end{bmatrix}$.

7. (Matlab/Octave Programming) Consider the Matlab function

```
function [L,U] = lu_cs210(A)
n = size(A,1);
L = zeros(size(A));
U = zeros(size(A));
A2 = A;
for k = 1:n
    if A2(k,k) == 0
        'Encountered 0 pivot. Stopping'
        return
    end
    for i = 1:n
        L(i,k) = A2(i,k)/A2(k,k);
        U(k,i) = A2(k,i);
    end
    for i = 1:n
        for j = 1:n
            A2(i,j) = A2(i,j) - L(i,k)*U(k,j);
        end
    end
end
```

(a) Try this code on the matrix `A = [ 10^-20 1; 1 1 ]`. What are L and U (You may need to access the elements as e.g., U(1,1) to see the values more accurately)?

(b) Try Matlab's `lu` function on the same matrix. What result do you get?

(c) Modify the code above to implement partial (row) pivoting. Try it on the matrix A above. What factors L and U do you get? What permutation? Try your code on `A2 = [ 1 2 3; 3 2 1; 1 -1 0]`. What do you get for L, U, and P? Compare with Matlab's `[L,U,P] = lu(A2)`. Include your code with your submission.

Solution: We call

```
[Lc,Uc] = lu_cs210(A)
```

(a)

$$Lc = \begin{pmatrix} 1 & 0 \\ 1e+20 & 1 \end{pmatrix}, \quad Uc = \begin{pmatrix} 1e-20 & 1 \\ 1.11022302462516e-16 & -1e+20 \end{pmatrix}$$

(b) Running `[L,U] = lu(A)`, we get

$$L = \begin{pmatrix} 1e-20 & 1 \\ 1 & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Matlab returns an upper triangular matrix, U, and a permutation of a lower triangular matrix, L, such that $A = LU$. The Matlab result uses the largest element in magnitude of each row at every step of the algorithm to do the elimination, and results in a more numerically stable algorithm. We can see that the backward error in the Matlab result is much smaller than backward error in this case:

```
>> norm(A-L*U)
ans =   0
>> norm(A-Lc*Uc)
ans =   1
```

We can also call this version of Matlab's `lu` function

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

to get unit lower triangular L, upper triangular U, and permutation matrix P such that $PA = LU$. Here

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

showing that the Matlab algorithm used the (2,1) entry as the first pivot.

(c) A simple extension of the code to include row pivoting is as follows:

```
function [L,U,P] = lu_cs210_pivot(A)
    n = size(A,1);
    L = zeros(size(A));
    U = zeros(size(A));
    A2 = A;

    for k = 1:n
        % find the biggest in magnitude
        [val,p] = max(abs(A2(:,k)));
        P(k) = p;

        if A2(p,k) == 0
            'Encountered 0 pivot.  Stopping'
            return
        end

        for i = 1:n
            L(i,k) = A2(i,k)/A2(p,k);
            U(k,i) = A2(p,i);
        end

        for i = 1:n
            for j = 1:n
                A2(i,j) = A2(i,j) - L(i,k)*U(k,j);
            end
        end
    end

    % permute L to be lower triangular
    L = L(P,:);
    % construct the permutation matrix
    I = eye(n);
    P = I(P,:);

end
```

Running this on the matrix A, we get

$$L = \begin{pmatrix} 1 & 0 \\ 10^{-20} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

which is the same result as Matlab gives.
For $A2$, running this code gives

$$L = \begin{pmatrix} 1.0000 & 0 & 0 \\ 0.3333 & 1.0000 & 0 \\ 0.3333 & -0.8000 & 1.0000 \end{pmatrix}, \quad U = \begin{pmatrix} 3.0000 & 2.0000 & 0000 \\ 0 & -1.6 & -0.3333 \\ 0 & 0 & 2.4000 \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

which is the same output as Matlab's `lu` produces.
More efficient as well as in-place implementations of lu are possible, for example, this implementation:

```
function [A,P] = lu_cs210_in_place(A)
    n = size(A,1);
    P = eye(n);
    for k = 1:n
        % find the biggest in magnitude
        [val,p] = max(abs(A(k:end,k)));
```

```
        p = p + k -1 ;
        % exchange rows
        if(p > k)
            temp = A(k,:);
            A(k,:) = A(p,:);
            A(p,:) = temp;
            temp = P(k,:);
            P(k,:) = P(p,:);
            P(p,:) = temp;
        end

        if A(k,k) == 0
            'Encountered 0 pivot.  Stopping'
            return
        end

        for i = k+1:n
            A(i,k) = A(i,k)/A(k,k);
        end

        for i = k+1:n
            for j = k+1:n
                A(i,j) = A(i,j) - A(i,k)*A(k,j);
            end
        end
    end

end
```