## Problem 1 (Question 2.2)

1. *PLU* factorization:
Since permuting the matrix does not change the number of arithmetic operations, assume the matrix is permuted properly.
For each column $i$, we need to

1. Determine $L_{ji}$ for $j = i+1 : n$: takes $n-i-1$ FLOP
2. Determine $A_{ii}$: takes $2(n-i-1)^2$ FLOP to perform multiplication and subtraction.

Hence, the total number of FLOPS needed for LPU factorization is

$$F_{LPU} = \sum_{i=n}^{2} 2(i-1)^2 + (i-1) = \frac{2}{3}n^3 + O(n^2)$$

Once the *LPU* factorization is done, we solve the system using froward and backward substitution.
A single round of forward substitution takes $n^2 + O(n)$ FLOPS (also true for backward substitution).
Since there are $m$ columns, the total number of FLOPS need for solving the system is

$$F_{solve} = mn^2 + O(mn)$$

Hence, the total number of FLOPS is given by

$$F = \frac{2}{3}n^3 + mn^2 + O(n^2) + O(mn)$$

2. Computing inverses:
To find the inverse, we use the Gauss Jordan method.
To convert the system from left to right side

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & 1 & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & a_{2n} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 & 0 & \ddots & 0 \\ a_{n1} & a_{n2} & \cdots & a_{nn} & 0 & 0 & \cdots & 1 \end{bmatrix} \to \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & 1 & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & a_{2n} & b_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & a_{nn} & b_{n1} & b_{n2} & \cdots & 1 \end{bmatrix}$$

It takes

$$\sum_{i=1}^{n-1} 2(n-i+1)(n-i) = \frac{2}{3}n^3 + O(n)$$

Clearing the upper half takes the same FLOPS, scaling the diagonal takes another $O(n)$ FLOPS. Hence, the total number of FLOPS to find the inverse is

$$F_{inverse} = \frac{4}{3}n^3 + O(n^2)$$

Finally, multiplying $A^{-1}$ with $B$

$$F_{matmul} = mn(2n-1)$$

FLOPS. Hence, the total numer of FLOPS needed for the inverse method is given by

$$F = \frac{2}{3}n^3 + 2mn^2 + O(n^2) + O(mn)$$

Hence, asymptotically, inverting the matrix takes longer time.

## Problem 2

1.
Let $P$ be a permutation matrix with $P_{i\sigma(i)} = 1$, where $\sigma(i)$ is a permutation in the symmetric group of order $n$.
Note that

$$[PX]_{ij} = \sum_k P_{ik}X_{kj} = X_{\sigma ij}$$

Hence, row $i$ is permuted to row $\sigma(i)$. Similarly,

$$[XP]_{ij} = \sum_k X_{ik}P_{kj} = X_{i\sigma^{-1}(j)}$$

Implying that column $j$ is permuted to column $\sigma^{-1}(j)$.

2.

$$\begin{aligned}[PP^T]_{ij} &= \sum_{k=1}^{n} P_{ik}P^T_{kj} \\ &= \sum_{k=1}^{n} P_{ik}P_{jk}\end{aligned}$$

Note that when $i \neq j$, $P_{ik}, P_{jk}$ cannot both be one, hence,

$$[PP^T]_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

3.
Since

$$\det(P) = \sum_\sigma \operatorname{sgn}(\sigma) p_{1\sigma(1)}...p_{n\sigma(n)}$$

Note that $p_{1\sigma(1)}...p_{n\sigma(n)} = 1$ if true only for one permutation $\sigma^*$, hence,

$$\det(P) = \operatorname{sgn}(\sigma^*) \cdot 1 = \pm 1$$

4.
Note that there is a bijection between permutation matrices and permutations $\sigma$ in a symmetric group.
Given a permutation matrix, suppose $P_{ik_i} = 1$, define $\sigma(i) = k_i$.
If we have two permutation matrices $P_1, P_2$, we can bijectively identify them as permutation $\sigma_1, \sigma_2$.
Since $P_1P_2$ is identified as $\sigma_1 \circ \sigma_2$, it follows that it is again a permutation matrix.

## Problem 3

Since $A = LDM^T$,

$$\det(A) \neq 0 \implies \det(L)\det(D)\det(M) \neq 0 \implies M^{-1} \text{ exists}$$

Then

$$M^{-1}A(M^{-1})^T = M^{-1}LD \tag{*}$$

Is still symmetric. Since the inverse of a lower triangular matrix is lower triangular, it follows that the RHS of $(*)$ is lower triangular.
Since it is also symmetric it must be the case that $M^{-1}LD$ is diagonal. Multiplying both sides by $D^{-1}$ suggests that $M^{-1}L$ is also diagonal.
Note that, however, multiplying these matrices gives diagonal elements are 1. This suggests that $M^{-1}L = I$, or $M = L$

## Problem 4

Consider the system

$$\begin{bmatrix} 1.001 & -1.999 \\ -2.001 & 6.001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.000 \\ 1.000 \end{bmatrix}$$

The true solution is give by

$$x = \begin{bmatrix} 999.750 \\ 500.125 \end{bmatrix}$$

Consider solving this using GEPP.
We assume three decimal floating point arithmetic.
We first use Matlab's `lu` function to factorize $A$:

$$A = \begin{bmatrix} 1.000 & 0 \\ -0.334 & 1.000 \end{bmatrix} \begin{bmatrix} -3.001 & 6.001 \\ 0 & 0.003 \end{bmatrix}$$

Solving the system using forward and backward substitution, we get

$$\begin{bmatrix} 1.000 & 0 \\ -0.334 & 1.000 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \implies \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1.000 \\ 1.334 \end{bmatrix}$$

$$\begin{bmatrix} 1.001 & 1.001 \\ 0 & -0.001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.000 \\ 1.334 \end{bmatrix} \implies \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 999.750 \\ 500.125 \end{bmatrix}$$

Hence,

$$\hat{x}_{GEPP} = \begin{bmatrix} 999.750 \\ 500.125 \end{bmatrix}$$

Which is perfectly accurate.

Now consider solve using Cramer's rule, we have

$$\det(A) = 1.001 \cdot 6.001 + 2.001 \cdot 1.999 = 0.007$$

Under three digit rounding. Hence,

$$x_1 = \frac{a_{22}b_1 - a_{12}b_2}{0.007} = 1142.857$$

$$x_2 = \frac{a_{22}b_1 - a_{12}b_2}{0.007} = 571.71$$

Hence,

$$\hat{x}_{Cramer} = \begin{bmatrix} 1142.857 \\ 571.71 \end{bmatrix}$$

Since

$$\frac{||\delta x_{Cramer}||}{||x_{Cramer}|||} = 0.143$$

Since we are using three digit floating point arithmetic, it follows that Cramer's rule is not backward stable in this example.

## Problem 5

Consider

$$X = \begin{bmatrix} I & -Z \\ 0 & I \end{bmatrix}$$

Then

$$XY = \begin{bmatrix} I & -Z \\ 0 & I \end{bmatrix} \begin{bmatrix} I & Z \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} = I$$

The other side follows the same argument. Hence

$$\begin{aligned} \kappa_F(Y) &= ||Y||_F||Y^{-1}||_F \\ &= ||Y||_F||X||_F \\ &= \sqrt{(2n + ||X||_F^2)}\sqrt{(2n + ||X||_F^2)} = 2n + ||Z||_F \end{aligned}$$

## Problem 6

Assume $m = n$, in Toledo's recursive LU algorithm, we have

$$A(n) \leq 2A(\frac{n}{2}) + O(m^3)$$

Since we call `RLU` on on two matrices of 1/2 of size.
Solving the lower traingular system

$$A_{12} = L_{11}U_{12}$$

Takes $O(m^3)$ FLOPS. Updating

$$A_{22} = A_{22} - L_{21}U_{12}$$

Also takes $O(m^3)$ FLOPS. This gives us the recurrence $(*)$.
Solving the recurrence using Master's theorem, we see that asymptotically,

$$A(n) = O(n^3)$$

We now count the number of words moved $W(n)$.
The recurrence is given by

$$W(n) = 4W(\frac{n}{2}) + 4 \cdot 3\left(\frac{n}{2}\right)^2$$

Since there are 4 matrices in the recursive calls, and 4 lines of code (line 8, 9, 11, 12), each involving 2 reads and 1 writes.
The recurrence stops when the matrices fit in the cache.
Solving this recurrence using the same way in lecture, we see that

$$W(n) = O(\frac{n^3}{\sqrt{M}})$$