Student Name: Seniha Serra Bozkurt
Student ID: 150190710
Class CRN: 21137

May 5, 2021
Istanbul Technical University
Department of Computer Engineering

BLG 202E - Numerical Methods in CE
Assignment 2

QUESTION 1 (15 pt.):

Problem:

$$\begin{aligned} x_1 - x_2 + 3x_3 &= 2 \\ x_1 + x_2 &= 4 \\ 3x_1 - 2x_2 + x_3 &= 1 \end{aligned}$$

Gaussian elimination in its simplest form.
Resulting **upper triangular matrix**.
Solution by **backward substitution.**

SOLUTION:

1. Write augmented matrix:

$$\left\{\begin{matrix} 1 & -1 & 3 & 2 \\ 1 & 1 & 0 & 4 \\ 3 & -2 & 1 & 1 \end{matrix}\right\}$$

2. Operation: Add $(-1).R_1$ to R_2 $\rightrightarrows$

$$\left\{\begin{matrix} 1 & -1 & 3 & 2 \\ 0 & 2 & -3 & 2 \\ 3 & -2 & 1 & 1 \end{matrix}\right\}$$

3. Operation: Add $(-3).R_1$ to R_3 $\rightrightarrows$

$$\left\{\begin{matrix} 1 & -1 & 3 & 2 \\ 0 & 2 & -3 & 2 \\ 0 & 1 & -8 & -5 \end{matrix}\right\}$$

4. Operation: Add $(-1/2).R_2$ to R_3 $\rightrightarrows$

$$\left\{\begin{matrix} 1 & -1 & 3 & 2 \\ 0 & 2 & -3 & 2 \\ 0 & 0 & -\frac{13}{2} & -6 \end{matrix}\right\}$$

So the resulting upper triangular matrix is:

$$\left\{\begin{matrix} 1 & -1 & 3 & 2 \\ 0 & 2 & -3 & 2 \\ 0 & 0 & -\frac{13}{2} & -6 \end{matrix}\right\}$$

Proceeding by **backward substitution:**

1. Start from 3rd row:
   $(-13/2).x_3 = -6$
   $\mathbf{x_3 = 12/13}$

2. Move 1 row up, to the 2nd row, put x_3 to its place:
   $2.x_2 + (-3).x_3 = 2$
   $2.x_2 + (-3).(12/13) = 2$
   $\mathbf{x_2 = 31/13}$

3. Move 1 row up, to the 1st row, put x_2 and x_3 to their places:
   $1.x_1 + (-1).x_2 + 3.x_3 = 1$
   $x_1 + (-1).(31/13) + 3.(12/13) = 1$
   $\mathbf{x_1 = 12/13}$

So, THE RESULTS:
$\mathbf{x_1 = 21/13}$
$\mathbf{x_2 = 31/13}$
$\mathbf{x_3 = 12/13}$

QUESTION 2 (25 pt.):

a. Matrix A =

$$\begin{Bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -2 \end{Bmatrix}$$

Since the 3rd row contains 3 zeros and 4th row contains 2 zeros, we need to replace them both to obtain a matrix A in **upper triangular** form. To do this job, we need a **P matrix** which is the **Identity Matrix**, but the R3 and R4 are **swapped** between each other. So that if we multiply this matrix P with A, we would obtain the matrix A with R3 and R4 **interchanged**.

So matrix P =

$$\begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{Bmatrix}$$

So that P.A = L.U =

$$\underbrace{\begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{Bmatrix}}_{P} \underbrace{\begin{Bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -2 \end{Bmatrix}}_{A} = \underbrace{\begin{Bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{Bmatrix}}_{L.U}$$

Since U is an upper triangular matrix, and L is a lower triangular matrix, while decomposing the matrix P.A, LU factorization works as:

- We should carry out **Gaussian** operations to make the P.A an upper triangular matrix
- While doing them, we should **record** our operations to an **identity matrix** on the left
- In the end, that new formed record matrix will become our **L matrix** and the remaining upper triangular matrix will be our **matrix U**.

1. When we write the 1st step of LU decomposition as following,
2. Since the P.A matrix is already an upper triangular matrix, this is the end.

$$\underset{L}{\begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{Bmatrix}} \Bigg| \underset{U}{\begin{Bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{Bmatrix}}$$

$$P = \begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{Bmatrix}, \quad A = \begin{Bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -2 \end{Bmatrix}$$

$$L = \begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{Bmatrix}, \quad U = \begin{Bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{Bmatrix}$$

PART - B: **Solution: A.x = b** $\Rightarrow$ **A^{-1}.A.x = A^{-1}.b** $\Rightarrow$ **x = A^{-1}.b**

But getting the inverse of A is hard. So I will come up with another solution, which includes the results I found in **part-a**:

A.x = b

P.A.x = P.b

I know that P.A = L.U ==>> So replace P.A 's with L.U:

L.U.x = P.b

Matrix visualisation:

$$\underset{\textbf{L}}{\begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{Bmatrix}} . \underset{\textbf{U}}{\begin{Bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{Bmatrix}} . \underset{\textbf{x}}{\begin{Bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{Bmatrix}} = \underset{\textbf{P}}{\begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{Bmatrix}} . \underset{\textbf{b}}{\begin{Bmatrix} 26 \\ 9 \\ 1 \\ -3 \end{Bmatrix}}$$

Since L is an identity matrix, any multiplication will result with the same, so we can eliminate it:

$$\begin{Bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{Bmatrix} . \begin{Bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{Bmatrix} = \begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{Bmatrix} . \begin{Bmatrix} 26 \\ 9 \\ 1 \\ -3 \end{Bmatrix}$$

U **x** **P** **b**

Multiplying any matrix with P will result in interchanging 3rd and 4th rows, as we discussed in part-a, this was matrix P's main purpose. So P.b =

$$\begin{Bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{Bmatrix} . \begin{Bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{Bmatrix} = \begin{Bmatrix} 26 \\ 9 \\ -3 \\ 1 \end{Bmatrix}$$

U **x** **P.b**

After this step, we know that **x4 = 1,** so we will continue with backward substitution:

(-1).x3 + (-2).x4 = -3

(-1).x3 + (-2).1 = -3

x3 = 1

4.x2 + 3.x3 + 2.x4 = 9

4.x2 + 3.1 + 2.1 = 9

x2 = 1

5.x1 + 6.x2 + 7.x3 + 8.x4 = 26

5.x1 + 6.1 + 7.1 + 8.1 = 26

x1 = 1

So resulting x matrix is:

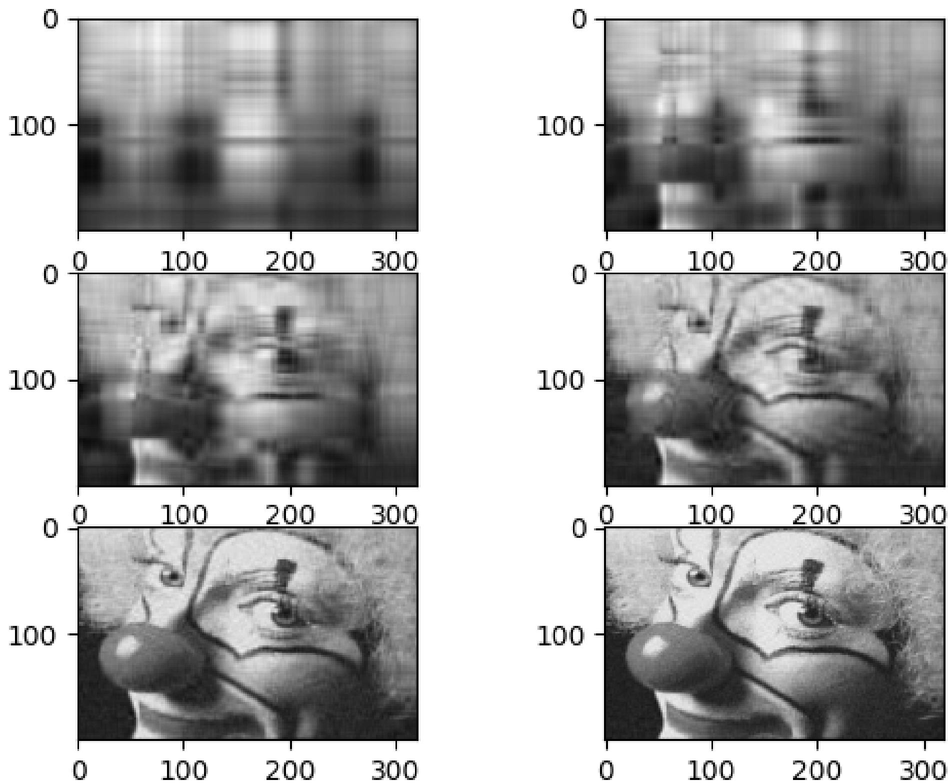
$$\begin{Bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix}$$

QUESTION 3 (30 pt.) SOLUTION: Code provided in the homework directory.
The output of the solution code: FIRST 5 ITERATION OF EACH OPERATION:

```
Operation starting from eigenvector V0-1 = [1 2 1]
Iteration: 1 estimated eigenvalue: 3.000000000000001

Iteration: 2 estimated eigenvalue: 3.0

Iteration: 3 estimated eigenvalue: 2.9999999999999996

Iteration: 4 estimated eigenvalue: 2.9999999999999996

Iteration: 5 estimated eigenvalue: 2.9999999999999996

EIGENVALUE - 1: 2.9999999999999996
EIGENVECTOR - 1: [0.57735027 0.57735027 0.57735027]


Operation starting from eigenvector V0-2 = [ 1  2 -1]
Iteration: 1 estimated eigenvalue: -4.714285714285715

Iteration: 2 estimated eigenvalue: -5.640000000000001

Iteration: 3 estimated eigenvalue: -5.90721649484536

Iteration: 4 estimated eigenvalue: -5.976623376623377

Iteration: 5 estimated eigenvalue: -5.994144437215356

EIGENVALUE - 2: -5.994144437215356
EIGENVECTOR - 2: [-0.69215012  0.0147266   0.72160331]
```

QUESTION 4 (30 pt.) SOLUTION: codes for both part-a and part-b are provided in the homework directory, as 4a.py and 4b.py. Outputs:

a. **Output: (photo also provided in the homework directory)**



b. When rank = 2, dimensions of U = [200, 2] S = [2] and V = [2, 320]. Required sizes: for **U = 200*2, V= 2*320, S= 2 => total = 200*2 + 2*320+2*1 = 2*(200+320+1)**

- When rank = 4, dimensions of U = [200, 4] S = [4] and V = [4, 320]. Required sizes: for **U = 200*4, V= 4*320, S= 4 => total = 200*4 + 4*320 +4*1= 4*(200+320+1)**
- When rank = 8, dimensions of U = [200, 8] S = [8] and V = [8, 320]. Required sizes: for **U = 200*8, V= 8*320, S= 8*1 => total = 200*8 + 2*320 = 8*(200 + 320 + 1)**
- When rank = 16, dimensions: U = [200, 16], S = [16] and V = [16, 320]. Required sizes: for **U = 200*16, V=16*320, S= 16*1=> total = 16*(200 + 320 + 1)**
- When rank = 32, dimensions: U = [200, 32] S = [32] and V = [32, 320]. Required sizes: for **U = 200*32, V= 32*320, S= 32*1 => total = 32*(200 + 320 + 1)**
- When rank = 64, dimensions: U = [200, 64] S = [64] and V = [64, 320]. Required sizes: for **U = 200*64, for V= 64*320, for S= 64*1 => total = 64*(200 + 320 + 1)**

So size = f(rank) = rank * (m + n + 1) , Proof can be seen as the output of the code above:

```
Original image dimensions (200, 320)

In this code we will view strucural similarities between original image
and compressed images as an indicator for performance of the truncated
SVD)
When compressed image's rank = 2
SHAPES: U: (200, 2), S: (2, 2), V: (2, 320)
Strucural similarity = 0.307601321008314
Size of the compressed image: 40.53125

When compressed image's rank = 4
SHAPES: U: (200, 4), S: (4, 4), V: (4, 320)
Strucural similarity = 0.3649603675084198
Size of the compressed image: 45.9990234375

When compressed image's rank = 8
SHAPES: U: (200, 8), S: (8, 8), V: (8, 320)
Strucural similarity = 0.5908453167511715
Size of the compressed image: 71.189453125

When compressed image's rank = 16
SHAPES: U: (200, 16), S: (16, 16), V: (16, 320)
Strucural similarity = 0.6771758547298351
Size of the compressed image: 76.07421875

When compressed image's rank = 32
SHAPES: U: (200, 32), S: (32, 32), V: (32, 320)
Strucural similarity = 0.7817877360813614
Size of the compressed image: 87.2939453125

When compressed image's rank = 64
SHAPES: U: (200, 64), S: (64, 64), V: (64, 320)
Strucural similarity = 0.8817262116541545
Size of the compressed image: 100.7998046875

As you can see from the shapes of U, S and V, the sizes of each U is m*r
and sizes of each S is r*1 and sizes of each V is r*n
Storage used for each image as a function of r: f(r) = r * (m + n + 1)
As you can see from the sizes of each compressed image, it gets increased
when the rank increases
```