Q1

a - b - c - d - e - f - g - h - i - j - k
2 0 2 2 0 8 0 8 0 6 9

20220808069
Burat Yalçın
20220808010
Mustafa Güvez
20220808005
Yahya Efe Kuruşa

x_i :	1	1,5	1,9	2,3	2,6	3
y_i :	2	2	7	10	35	56

a) find the least squares regression line for the given data using QR factorization of the design matrix

$y = ax + b \Rightarrow$
$a + b = 2 \qquad 2{,}3a + b = 10$
$1{,}5a + b = 2 \qquad 2{,}6a + b = 35$
$1{,}9a + b = 7 \qquad 3a + b = 56$

$Ax = b$

$$A = \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 1{,}5 & 1 & 2 \\ 1{,}9 & 1 & 7 \\ 2{,}3 & 1 & 10 \\ 2{,}6 & 1 & 35 \\ 3 & 1 & 56 \end{array}\right] \quad a_1 = \begin{bmatrix} 1 \\ 1{,}5 \\ 1{,}9 \\ 2{,}3 \\ 2{,}6 \\ 3 \end{bmatrix} \quad a_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$a_1, a_2 = 1 + 1{,}5 + 1{,}9 + 2{,}3 + 2{,}6 + 3$
$= 12{,}3 \neq 0$

They are not ortagonal so, we need to make them ortagonal, let's use Gram - Schmidt method.

$v_1 = a_1$

$$v_2 = a_2 - \frac{a_2 \cdot v_1}{v_1 \cdot v_1} v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \frac{12{,}3}{27{,}91} \begin{bmatrix} 1 \\ 1{,}5 \\ 1{,}9 \\ 2{,}3 \\ 2{,}6 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 - 0{,}4407 \\ 1 - 0{,}66105 \\ 1 - 0{,}83733 \\ 1 - 1{,}01361 \\ 1 - 1{,}14582 \\ 1 - 1{,}3221 \end{bmatrix}$$

$v_1 \cdot v_1 = 1 + (1{,}5)^2 + (1{,}9)^2 + (2{,}3)^2 + (2{,}6)^2 + 3^2 = 27{,}91$

$\|v_1\| = 5{,}2829$

$v_2 \cdot v_2 = (0{,}5593)^2 + (0{,}33895)^2 + (0{,}16267)^2 + (0{,}1361)^2 + (0{,}14582)^2 + (0{,}3221)^2 \approx 0{,}5977$

$$v_2 = \begin{bmatrix} 0{,}5593 \\ 0{,}33895 \\ 0{,}16267 \\ -0{,}1361 \\ -0{,}14582 \\ -0{,}3221 \end{bmatrix} \qquad e_1 = \frac{v_1}{\|v_1\|} = \begin{bmatrix} 0{,}18928 \\ 0{,}28392 \\ 0{,}35963 \\ 0{,}48534 \\ 0{,}49213 \\ 0{,}56784 \end{bmatrix}$$

$$e_2 = \frac{v_2}{\|v_2\|} = \begin{bmatrix} 0{,}73482 \\ 0{,}44531 \\ 0{,}21372 \\ -0{,}01788 \\ -0{,}19157 \\ -0{,}42315 \end{bmatrix} \qquad Q = \begin{bmatrix} 0{,}18928 & 0{,}73482 \\ 0{,}28392 & 0{,}44531 \\ 0{,}35963 & 0{,}21372 \\ 0{,}43534 & -0{,}01788 \\ 0{,}49213 & -0{,}19157 \\ 0{,}56784 & -0{,}42315 \end{bmatrix}$$

$A = 6 \times 2 \xrightarrow{2} R = 2 \times 2$

$$R = \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix} \qquad r_{11} = a_1 \cdot e_1 \qquad r_{12} = a_2 \cdot e_1 \qquad r_{22} = a_2 \cdot e_2$$

$$r_{11} = a_1 \cdot e_1 = 1 \cdot 0{,}18928 + 1{,}5 \cdot 0{,}28392 + 1{,}9 \cdot 0{,}35963 + 2{,}3 \cdot 0{,}43534 + 2{,}6 \cdot 0{,}49213 + 3 \cdot 0{,}56784$$

$$= 5{,}282797$$

$$r_{12} = a_2 \cdot e_1 = 1 \cdot 0{,}18928 + 0{,}28392 + 0{,}35963 + 0{,}43534 + 0{,}49213 + 0{,}56784$$

$$= 2{,}32814$$

$$r_{22} = a_2 \cdot e_2 = 0{,}73482 + 0{,}44531 + 0{,}21372 - 0{,}01788 - 0{,}19157 - 0{,}42315$$

$$= 0{,}76116$$

$$R = \begin{bmatrix} 5{,}282797 & 2{,}32814 \\ 0 & 0{,}76116 \end{bmatrix}, \qquad R^{-1} = \frac{1}{\underbrace{\det(R)}_{4{,}02105}} \begin{bmatrix} 0{,}76116 & -2{,}32814 \\ 0 & 5{,}282797 \end{bmatrix}$$

$$X = \underbrace{R^{-1} \underbrace{Q^T b}_{first}}_{second} \qquad = \begin{bmatrix} 0{,}18929 & -0{,}57898 \\ 0 & 1{,}31378 \end{bmatrix}$$

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{bmatrix} 0{,}18929 & -0{,}57898 \\ 0 & 1{,}31378 \end{bmatrix} \cdot \begin{bmatrix} 56{,}8408 \\ -26{,}72385 \end{bmatrix}$$

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 26{,}23196 \\ -35{,}10925 \end{pmatrix}, \qquad y \approx 26{,}23196\,x - 35{,}10925$$

Solution section added with both numpy library and Scipy library. There is a little difference between the results.

20220808069 Burak Yalçın

20220808010 Mustafa GÜNEŞ

20220808005 Yahya Efe KURUŞAY

Question1a.java Code and Output

```python
import numpy as np
import scipy.linalg

# Öğrenci ID: 20220808069 -> a=2, c=2, j=6, k=9
a_id = 2
c_id = 2
j_id = 6
k_id = 9

x_i = np.array([1, 1.5, 1.9, 2.3, 2.6, 3])
y_i_formulas = [
    c_id,
    a_id,
    j_id + 1,
    k_id + 1,
    (k_id + 1) * (j_id + 1) / 2,
    k_id * j_id + 2
]
y_i = np.array(y_i_formulas)

print("Data Points:")
for i in range(len(x_i)):
    print(f"x_{i+1} = {x_i[i]}, y_{i+1} = {y_i[i]}")
print("-" * 30)

A = np.vstack([x_i, np.ones(len(x_i))]).T

print("Design Matrix A:")
print(A)
print("-" * 30)

print("y Vektörü:")
print(y_i)
print("-" * 30)

# Solution 1: using numpy.linalg.qr
print("QR Decomposition with NumPy:")
Q_np, R_np = np.linalg.qr(A)

print("Q (NumPy):")
print(Q_np)
print("\nR (NumPy):")
print(R_np)
print("-" * 30)
```

```python

# Q.T * y
qty_np = Q_np.T @ y_i
print("Q.T @ y (NumPy):")
print(qty_np)
print("-" * 30)

# R * beta = qty_np
# beta = [beta1, beta0]
beta_np = np.linalg.solve(R_np, qty_np)
beta1_np = beta_np[0]
beta0_np = beta_np[1]

print(f"Coefficients (beta) found with NumPy: {beta_np}")
print(f"Regression Line (NumPy): y = {beta1_np:.4f}x + {beta0_np:.4f}")
print(f"Our Manuel Solution with Calculator :    y = 26.2338x - 35.1097")
print("-" * 50)

# Solution 2: using scipy.linalg.qr
print("QR Decomposition with SciPy:")
Q_sp, R_sp = scipy.linalg.qr(A, mode='economic')

print("Q (SciPy):")
print(Q_sp)
print("\nR (SciPy):")
print(R_sp)
print("-" * 30)

# R * beta = Q.T * y to be solved
qty_sp = Q_sp.T @ y_i
print("Q.T @ y (SciPy):")
print(qty_sp)
print("-" * 30)

beta_sp = scipy.linalg.solve_triangular(R_sp, qty_sp, lower=False)
beta1_sp = beta_sp[0]
beta0_sp = beta_sp[1]

print(f"Coefficients found with SciPy (beta): {beta_sp}")
print(f"Regression Line (SciPy): y = {beta1_sp:.4f}x + {beta0_sp:.4f}")
print("-" * 50)

#Check directly with numpy.linalg.lstsq
print("Direct Solution (Control) with NumPy lstsq:")
beta_lstsq, residuals, rank, s_values = np.linalg.lstsq(A, y_i, rcond=None)
```

```python

#Check directly with numpy.linalg.lstsq
print("Direct Solution (Control) with NumPy lstsq:")
beta_lstsq, residuals, rank, s_values = np.linalg.lstsq(A, y_i, rcond=None)
beta1_lstsq = beta_lstsq[0]
beta0_lstsq = beta_lstsq[1]

print(f"Coefficients found with lstsq (beta): {beta_lstsq}")
print(f"Regression Line (lstsq): y = {beta1_lstsq:.4f}x + {beta0_lstsq:.4f}")
print("-" * 50)
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\burak\OneDrive\Masaüstü\lineer\Assignment2>
& C:/Users/burak/AppData/Local/Programs/Python/Python312/python.exe c:/Users/burak/OneDrive/Masaüstü/lineer/Assignment2/Question1a.py
Data Points:
x_1 = 1.0, y_1 = 2.0
x_2 = 1.5, y_2 = 2.0
x_3 = 1.9, y_3 = 7.0
x_4 = 2.3, y_4 = 10.0
x_5 = 2.6, y_5 = 35.0
x_6 = 3.0, y_6 = 56.0
------------------------------
Design Matrix A:
[[1.  1. ]
 [1.5 1. ]
 [1.9 1. ]
 [2.3 1. ]
 [2.6 1. ]
 [3.  1. ]]
------------------------------
y Vektörü:
[ 2.  2.  7. 10. 35. 56.]
------------------------------
QR Decomposition with NumPy:
Q (NumPy):
[[-0.18928669 -0.73479802]
 [-0.28393004 -0.4453036 ]
 [-0.35964472 -0.21370807]
 [-0.43535939  0.01788746]
 [-0.4921454   0.19158411]
 [-0.56786008  0.42317964]]

R (NumPy):
[[-5.28299158 -2.32822631]
 [ 0.         -0.76115848]]
```

```
------------------------------
Q.T @ y (NumPy):
[-56.84279364  26.72621841]
------------------------------
Coefficients (beta) found with NumPy: [ 26.23376623 -35.11255411]
Regression Line (NumPy): y = 26.2338x + -35.1126
Our Manuel Solution with Calculator :    y = 26.23196x - 35.10925
--------------------------------------------------
QR Decomposition with SciPy:
Q (SciPy):
[[-0.18928669 -0.73479802]
 [-0.28393004 -0.4453036 ]
 [-0.35964472 -0.21370807]
 [-0.43535939  0.01788746]
 [-0.4921454   0.19158411]
 [-0.56786008  0.42317964]]

R (SciPy):
[[-5.28299158 -2.32822631]
 [ 0.         -0.76115848]]
------------------------------
Q.T @ y (SciPy):
[-56.84279364  26.72621841]
------------------------------
Coefficients found with SciPy (beta): [ 26.23376623 -35.11255411]
Regression Line (SciPy): y = 26.2338x + -35.1126
--------------------------------------------------
Direct Solution (Control) with NumPy lstsq:
Coefficients found with lstsq (beta): [ 26.23376623 -35.11255411]
Regression Line (lstsq): y = 26.2338x + -35.1126
--------------------------------------------------
```

We will add java codes as a zip file.

b) Use gradient method to compute same least squares regression line

Initial point: $x_0 = (0,0)$ or $x_0(1,1)$ or any randomly generated point.

learning rate: $\alpha = 0{,}001$

$\varepsilon = 0{,}005$

stopping criterian $\|\nabla f\| < \varepsilon$

x_i	1	1,5	1,9	2,3	2,6	3
y_i	2	2	7	10	35	56

$y = ax + b$

we will use gradient discent method

$$f(b,a) = \sum_{i=1}^{6} (y_i - b - a \cdot x_i)^2$$

, to solve this problem, firstly we need to do initializon

lets choose $x_0 = (0,0)$

$\alpha = 0{,}001 \quad \varepsilon = 0{,}005$

Then,
$$\left.\begin{aligned} x_1 &= x_0 - \alpha \cdot \nabla f(x_0) \\ x_2 &= x_1 - \alpha \cdot \nabla f(x_1) \\ &\vdots \\ x_n &= x_{n-1} - \alpha \cdot \nabla f(x_{n-1}) \end{aligned}\right\}$$

When $\|\nabla f(x)\| < \varepsilon$ we stop, and x_n is the approximate value for global minimum.

It is hard to solve manually. So, we write a python code to solve (we added the code block and output at the end)

According to result of code, $a = 26{,}2317$

$b = -35{,}1078$

$$y = 26{,}2317x - 35{,}1078$$

20220808068
Burak Yalçın
20220808010
Mustafa Güneş
20220808005
Yahya Efe Kuruçay

Question1b.java Code and Output

Question1b.py

```python
import numpy as np

# x_i = [1, 1.5, 1.9, 2.3, 2.6, 3]
# y_i = [2, 2, 7, 10, 35, 56]
data_x = np.array([1, 1.5, 1.9, 2.3, 2.6, 3])
data_y = np.array([2., 2., 7., 10., 35., 56.])

alpha = 0.001
epsilon = 0.005
max_iterations = 500000

beta = np.array([0.0, 0.0]) # beta[0] = beta_0 (intercept), beta[1] = beta_1 (slop

print(f"Başlangıç Katsayıları (beta0, beta1): {beta}")
print(f"Öğrenme Oranı (alpha): {alpha}")
print(f"Durdurma Toleransı (epsilon): {epsilon}")
print("-" * 50)

# To store the iteration history
cost_history = []
grad_norm_history = []

# Gradient Decrease Cycle
for iteration in range(max_iterations):
    beta0_current = beta[0]
    beta1_current = beta[1]

    # Predicted y values: y_pred = beta1*x + beta0
    y_pred = beta1_current * data_x + beta0_current

    # Hata terimleri: hata = y_pred - y_actual
    errors = y_pred - data_y

    # Cost Function (Sum of Squared Errors) - For follow-up
    cost = np.sum(errors**2)
    cost_history.append(cost)

    # Calculation of Gradients
    # ∂f/∂β₀ = 2 * Σ (errors)
    # ∂f/∂β₁ = 2 * Σ (errors * data_x)
    grad_beta0 = 2 * np.sum(errors)
    grad_beta1 = 2 * np.sum(errors * data_x)
    gradient = np.array([grad_beta0, grad_beta1])

    # Norm of the gradient - for the stopping criterion
    grad_norm = np.linalg.norm(gradient)
    grad_norm_history.append(grad_norm)

    # Stop Criteria
    if grad_norm < epsilon:
        print(f"\n*** Stop criterion met! ({iteration+1}. iteration) ***")
        break

    # Updating Coefficients
    beta = beta - alpha * gradient

    # Print progress status occasionally (optional)
    if (iteration + 1) % 50000 == 0 or iteration < 5:
        print(f"Iter {iteration+1:6}: β0={beta[0]:.4f}, β1={beta[1]:.4f}, ||∇f||={


if iteration == max_iterations - 1 and grad_norm >= epsilon:
    print(f"\nUyarı: Maximum number of iterations ({max_iterations}) ulaşıldı.")
    print(f"The gradient norm ({grad_norm:.6f}) is still greater than or equal to

print("-" * 50)
print("Gradient Descent Method Result:")
print(f"Iteration Number   : {iteration + 1}")
print(f"beta0 (intercept)  : {beta[0]:.4f}")
print(f"beta1 (slope)      : {beta[1]:.4f}")
print(f"Final Gradient Norm  : {grad_norm:.6f} (Hedef < {epsilon})")
print(f"The Last Mistake (Cost)    : {cost:.2f}")
print(f"Regression Line  : y = {beta[1]:.4f}x + {beta[0]:.4f}")
print("-" * 50)

# The result of (a) for comparison
beta1_qr = 26.23378378
beta0_qr = -35.10972973
print("Comparison (Problem 1a - QR/lstsq):")
print(f"beta0 (intercept)  : {beta0_qr:.4f}")
print(f"beta1 (slope)      : {beta1_qr:.4f}")
print(f"Regresyon Line  : y = {beta1_qr:.4f}x + {beta0_qr:.4f}")
print("-" * 50)
```

Ln 71, Col 28 Spaces: 4 UTF-8 CRLF {} Python 3.12.0

Question1b.py


```python
import numpy as np

# x_i = [1, 1.5, 1.9, 2.3, 2.6, 3]
# y_i = [2, 2, 7, 10, 35, 56]
data_x = np.array([1, 1.5, 1.9, 2.3, 2.6, 3])
data_y = np.array([2., 2., 7., 10., 35., 56.])

alpha = 0.001
epsilon = 0.005
max_iterations = 500000

beta = np.array([0.0, 0.0]) # beta[0] = beta_0 (intercept), beta[1] = beta_1 (slop
```


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\burak\OneDrive\Masaüstü\lineer\Assignment2>
& C:/Users/burak/AppData/Local/Programs/Python/Python312/python.exe c:/Users/burak/OneDrive/Masa
Başlangıç Katsayıları (beta0, beta1): [0. 0.]
Öğrenme Oranı (alpha): 0.001
Durdurma Toleransı (epsilon): 0.005
--------------------------------------------------
Iter      1: β0=0.2240, β1=0.6006, ||∇f||=641.011981, Cost=4518.00
Iter      2: β0=0.4305, β1=1.1622, ||∇f||=598.340939, Cost=4120.78
Iter      3: β0=0.6208, β1=1.6873, ||∇f||=558.535174, Cost=3774.68
Iter      4: β0=0.7958, β1=2.1784, ||∇f||=521.404001, Cost=3473.09
Iter      5: β0=0.9567, β1=2.6379, ||∇f||=486.769534, Cost=3210.26

*** Stop criterion met! (9321. iteration) ***
--------------------------------------------------
Gradient Descent Method Result:
Iteration Number   : 9321
beta0 (intercept)  : -35.1078
beta1 (slope)      : 26.2317
Final Gradient Norm  : 0.004998 (Hedef < 0.005)
The Last Mistake (Cost)    : 572.61
Regression Line  : y = 26.2317x + -35.1078
--------------------------------------------------
Comparison (Problem 1a - QR/lstsq):
beta0 (intercept)  : -35.1097
beta1 (slope)      : 26.2338
Regresyon Line  : y = 26.2338x + -35.1097
--------------------------------------------------
PS C:\Users\burak\OneDrive\Masaüstü\lineer\Assignment2>
```

20220808069
Burak Yalçın
20220808010
Mustafa GÜNEY
20220808005
Yahya Efe KURUŞAY

Q_2 / 20220808069

$J=6,\ k=9$

Consider the following 4x4 linear system.

$$17x_1 - x_2 + 2x_3 + x_4 = 8$$
$$x_1 + 20x_2 - 3x_3 - x_4 = 10$$
$$2x_1 - x_2 + 13x_3 - 2x_4 = 10$$
$$x_1 + 2x_2 + 4x_3 + 10x_4 = 18$$

In this problem, your goal is to approximately solve this system using Jacobi or Gauss-Seidel iteration.

Initial guess : $x_0 = (0,0,0)^T$

Stopping criterion : $XTOL = ||x_n - x_{n-1}|| < \varepsilon$ where $||\cdot||$ is the maximum norm.

$\varepsilon = 0{,}01$

$Ax = b$

$$A = \left[\begin{array}{cccc|c} 17 & -1 & 2 & 1 & 8 \\ 1 & 20 & -3 & -1 & 10 \\ 2 & -1 & 13 & -2 & 0 \\ 1 & 2 & 4 & 10 & 18 \end{array}\right]$$

$\rightarrow$ Since the absolute largest values in each column are on the diagonal, we can start to leave those values alone

$$17x_1 = 8 + x_2 - 2x_3 - x_4 \Rightarrow x_1^{(k+1)} = \frac{1}{17}(8 + x_2^{(k)} - 2x_3^{(k)} - x_4^{(k)})$$

$$x_2^{(k+1)} = \frac{1}{20}\cdot(10 - x_1^{(k)} + 3x_3^{(k)} + x_4^{(k)})$$

$$x_3^{(k+1)} = \frac{1}{13}\cdot(-2x_1^{(k)} + x_2^{(k)} + 2x_4^{(k)})$$

$$x_4^{(k+1)} = \frac{1}{10}(18 - x_1^{(k)} - 2x_2^{(k)} - 4x_3^{(k)})$$

$x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, x_4^{(0)} = 0$

Iteration 1: $(k=0,\ x_n^{(0)})$

$$x_1^{(1)} = \frac{1}{17}(8 + 0 - 2\cdot 0 - 0) = \frac{8}{17} \approx 0{,}470588$$

$$x_2^{(1)} = \frac{1}{20}(10 - 0 + 3\cdot 0 + 0) = \frac{10}{20} = 0{,}5$$

$$x_3^{(1)} = \frac{1}{13}(0) = \frac{0}{13} = 0$$

$$x_4^{(1)} = \frac{1}{10}(18) = \frac{18}{10} = 1{,}8$$

$$XTOL_1 = \max\{|0{,}470588 - 0|, |0{,}5 - 0|, |0 - 0|, |1{,}8 - 0|\}$$

$XTOL_1 = 1{,}8$

$1{,}8 \not< 0{,}01$ continue..

Iteration 2

$$x_1^{(2)} = \frac{1}{17}(8 + 0{,}5 - 2.0 - 1.8) = \frac{1}{17}\, 6{,}7 \approx 0.394118$$

$$x_2^{(2)} = \frac{1}{20}(10 - 0{,}470588 + 3.0 + 1{,}8) = \frac{1}{20} \cdot 11{,}329412 \approx 0.566471$$

$$x_3^{(2)} = \frac{1}{13}($$

$$x_4^{(2)} = \frac{1}{10}(18 - 0{,}470588 - 2.0{,}5 - 4.0) \approx 1{,}652941$$

$$XTOL_2 = \max\{|0{,}394118 - 0{,}470588|, |0{,}566471 - 0{,}5|, |0{,}242986 - 0|, |1{,}659241 - 1{,}8|\}$$

$$XTOL_2 = 0{,}242986 \geq 0{,}01 \quad \text{Continue.}$$

Iteration 3

$$x_1^{(3)} = \frac{1}{17}(8 + 0{,}566471 - 2.0{,}242986 - 1{,}652941) = \frac{1}{17} \cdot 6{,}427558$$

$$\approx 0{,}378092$$

$$x_2^{(3)} = \frac{1}{20}(10 - 0{,}394118 + 3 \cdot 0{,}242986 + 1{,}652941) = \frac{1}{20} \cdot (10 - 0{,}394118 + 0{,}728958 +$$

$$1{,}652941) = \frac{1}{20} \cdot (11{,}987781) \approx 0{,}599389$$

$$x_3^{(3)} = \frac{1}{13}(-2 \cdot 0{,}394118 + 0{,}566471 + 2 \cdot 1{,}652941) = \frac{1}{13}(3{,}084117) \approx 0{,}237240$$

$$x_4^{(3)} = \frac{1}{10}(18 - 0{,}394118 - 2 \cdot 0{,}566471 - 4 \cdot 0{,}242986) = \frac{1}{10}(15{,}501996) \approx 1{,}5501$$

$$XTOL_3 = 0{,}102481 \geq 0{,}01 \quad \text{Continue}$$

20220808069
Burak Yalçın
20220808010
Mustafa GÜVEZ
20220808005
Yahya Efe KURUŞAY

Iteration 4

$x_1^{(4)} = \frac{1}{17} \cdot (8 + 0{,}599389 - 2 \cdot 0{,}237240 - 1{,}550100) = \frac{1}{17} \cdot 6{,}574809$

$\approx 0{,}386753$

$x_2^{(4)} = \frac{1}{20} (10 - 0{,}378092 + 3 \cdot 0{,}237240 + 1{,}5501)$

$= \frac{1}{20} (11{,}883728) \approx 0{,}594186$

$x_3^{(4)} = \frac{1}{13} \cdot (-2 \cdot 0{,}378092 + 0{,}599389 + 2 \cdot 1{,}5501) = \frac{1}{13} \cdot (2{,}943405) \approx 0{,}226416$

$x_4^{(4)} = \frac{1}{10} (18 - 0{,}378092 - 2 \cdot 0{,}599389 - 4 \cdot 0{,}237240) = \frac{1}{10} (15{,}47417) \approx 1{,}547417$

$XTOL_4 = \max \{ |0{,}386753 - 0{,}378092|, |0{,}594186 - 0{,}599389|, |0{,}226416 - 0{,}237240|, |1{,}547417 - 1{,}5501| \}$

$XTOL_4 = 0{,}010824 \geq 0{,}01$ Continue. (I think, we're close :))

Iteration 5

$x_1^{(5)} = \frac{1}{17} (8 + 0{,}594186 - 2 \cdot 0{,}226416 - 1{,}547417 = \frac{1}{17} (6{,}593937) \approx 0{,}387879$

$x_2^{(5)} = \frac{1}{20} (10 - 0{,}386753 + 3 \cdot 0{,}226416 + 1{,}547417) = \frac{1}{20} \cdot 11{,}839912 \approx 0{,}591896$

$x_3^{(5)} = \frac{1}{13} (-2 \cdot 0{,}386753 + 0{,}594186 + 2 \cdot 1{,}547417) = \frac{1}{13} (2{,}915514) \approx 0{,}224270$

$x_4^{(5)} = \frac{1}{10} (18 - 0{,}386753 - 2 \cdot 0{,}594186 - 4 \cdot 0{,}226416) = \frac{1}{10} \cdot 15{,}519211 \approx 1{,}5519211$

$XTOL_5 = \max \{ |0{,}387899 - 0{,}386753|, |0{,}591896 - 0{,}594186|, |0{,}224270 - 0{,}226416|, |1{,}5519221 - 1{,}547417| \}$

$XTOL_5 = 0{,}004504 < 0{,}01$ Stop.

$x_n = x^5 : (0{,}387879 , 0{,}591996 , 0{,}224270 , 1{,}551921)$

$XTOL \approx 0{,}004504$

we use 6 digit

20220808069
Burak Yalçın
20220808010
Mustafa GÜNEŞ
20220808005
Yahya Efe KURUCAY

Question2.java Code and Output

```python
import numpy as np

j = 6
k_id = 9 # k can be a keyword in Python, so I used k_id

A = np.array([
    [11 + j, -1, 2, 1],
    [1, 11 + k_id, -3, -1],
    [2, -1, 19 - j, -2],
    [1, 2, 4, 19 - k_id]
], dtype=float)

b = np.array([8, 10, 0, 18], dtype=float)

print("A Matrisi:\n", A)
print("\nb Vektörü:\n", b)

# cInitial parameters
x_old = np.array([0.0, 0.0, 0.0, 0.0]) # x^(0)
epsilon = 0.01
max_iterations = 100
n = len(b)

print(f"\nInitial x^(0): {x_old}")
print(f"Epsilon: {epsilon}")
print("-" * 30)
print("-" * 30)

for iteration in range(max_iterations):
    x_new = np.zeros_like(x_old)

    # Apply Jacobi iteration formulae
    # x_new[0] (x1)
    x_new[0] = (b[0] - A[0,1]*x_old[1] - A[0,2]*x_old[2] - A[0,3]*x_old[3]) / A[0,
    # x_new[1] (x2)
    x_new[1] = (b[1] - A[1,0]*x_old[0] - A[1,2]*x_old[2] - A[1,3]*x_old[3]) / A[1,
    # x_new[2] (x3)
    x_new[2] = (b[2] - A[2,0]*x_old[0] - A[2,1]*x_old[1] - A[2,3]*x_old[3]) / A[2,
    # x_new[3] (x4)
    x_new[3] = (b[3] - A[3,0]*x_old[0] - A[3,1]*x_old[1] - A[3,2]*x_old[2]) / A[3,

    # Durdurma kriteri: xTOL = ||x_new - x_old||_∞
    xTOL = np.max(np.abs(x_new - x_old))
    print(f"Iteration {iteration + 1}:")
    print(f"  x_new = {np.round(x_new, 6)}")
    print(f"  xTOL  = {xTOL:.6f}")
```

```python
    print(f"  x_new = {np.round(x_new, 6)}")
    print(f"  xTOL  = {xTOL:.6f}")

    if xTOL < epsilon:
        print("-" * 30)
        print(f"Convergence{iteration + 1}. provided in iteration.")
        x_final = x_new
        xTOL_final = xTOL
        break

    x_old = x_new.copy() # Bir sonraki iterasyon için x_old'u güncelle
    print("-" * 30)

    if iteration == max_iterations - 1:
        print("-" * 30)
        print("Maximum number of iterations reached, no convergence achieved.")
        x_final = x_new
        xTOL_final = xTOL

print("\nConclusion:")
print(f"Final approximate solution x_n: {np.round(x_final, 6)}")
print(f"The xTOL value that caused the stop: {xTOL_final:.6f}")
# Kontrol için (opsiyonel): Ax - b ne kadar yakın sıfıra?
residual = np.dot(A, x_final) - b
print(f"\nKalıntı (Ax_n - b): {np.round(residual, 6)}")
```

```
------------------------------
Iteration 5:
  x_new = [0.387879 0.591996 0.22427  1.551921]
  xTOL  = 0.004504
------------------------------
  x_new = [0.387879 0.591996 0.22427  1.551921]
  xTOL  = 0.004504
------------------------------
Convergence5. provided in iteration.

  xTOL  = 0.004504
------------------------------
```