МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут прикладного системного аналізу

Кафедра штучного інтелекту

Звіт

про виконання лабораторної роботи №2 з дисципліни «Обчислювальна математика»

Виконав:

студент ІІ курсу, групи КІ-32

Присяжнюк Владислав

Прийняв:

доцент Квітка О. О.

Київ – 2024

Варіант 38

Вар. 38	44,9	3,38	-3	1	-1,1	0,5	-207,84
	1,38	70,9	-2,8	2,4	-4,4	-3,2	-383,98
	4,38	3,2	76,9	0,2	4,2	-3,4	223,64
	2,38	3,3	-3,2	88,4	-4,6	4,2	-146,3
	3,38	-1,2	-3,2	0,5	43,9	0,5	21,61
	1,38	4,3	1,5	-4,3	-0,2	58,4	117,81

Рис 1.1 – Індивідуальне завдання за варіантом

$$\begin{cases} 44.9x_1 + 3.38x_2 - 3x_3 + 1x_4 - 1.1x_5 + 0.5x_6 = -207.84 \\ 1.38x_1 + 70.9x_2 - 2.8x_3 + 2.4x_4 - 4.4x_5 - 3.2x_6 = -383.98 \\ 4.38x_1 + 3.2x_2 + 76.9x_3 + 0.2x_4 + 4.2x_5 - 3.4x_6 = 223.64 \\ 2.38x_1 + 3.3x_2 - 3.2x_3 + 88.4x_4 - 4.6x_5 + 4.2x_6 = -146.3 \\ 3.38x_1 - 1.2x_2 - 3.2x_3 + 0.5x_4 + 43.9x_5 + 0.5x_6 = 21.61 \\ 1.38x_1 + 4.3x_2 + 1.5x_3 - 4.3x_4 - 0.2x_5 + 58.4x_6 = 117.81 \end{cases}$$

$$\begin{bmatrix} 44.9 & 3.38 & -3 & 1 & -1.1 & 0.5 & -207.84 \\ 1.38 & 70.9 & -2.8 & 2.4 & -4.4 & -3.2 & -383.98 \\ 4.38 & 3.2 & 76.9 & 0.2 & 4.2 & -3.4 & 223.65 \\ 2.38 & 3.3 & -3.2 & 88.4 & -4.6 & 4.2 & -146.3 \\ 3.38 & -1.2 & -3.2 & 0.5 & 43.9 & 0.5 & 21.61 \\ 1.38 & 4.3 & 1.5 & -4.3 & -0.2 & 58.4 & 117.81 \end{bmatrix}$$

Розв’язок:

1. Перевіримо умову збіжності ітераційного процесу (діагональне домінування в матриці системи):

$$|44.9| \geq |3.38| + |-3| + |1| + |-1.1| + |0.5|,$$

$$|70.9| \geq |1.38| + |-2.8| + |2.4| + |-4.4| + |-3.2|,$$

$$|76.9| \geq |4.38| + |3.2| + |0.2| + |4.2| + |-3.4|,$$

$$|88.4| \geq |2.38| + |3.3| + |-3.2| + |-4.6| + |4.2|,$$

$$|43.9| \geq |3.38| + |-1.2| + |-3.2| + |0.5| + |0.5|,$$

$$|58.4| \geq |1.38| + |4.3| + |1.5| + |-4.3| + |-0.2|$$

Матриця задовольняє умові діагонального домінування.

Частина 1

Знайти розв'язки системи лінійних алгебраїчних рівнянь точними методами:

a. Крамера;

b. оберненої матриці;

c. LU – декомпозиції.

Перевірити правильність визначення коренів підстановкою.

Метод Крамера

Для обрахунків використаємо мову Python та бібліотеку NumPy що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами.

Почнемо з обрахунків визначника матриці коєфіцієнтів:

Для цього використаємо функцію np.linalg.det() яка працює для квадратних матриць та використовує LU-розклад матриці та обчислює визначник за допомогою добутку визначників матриць L та U:

$$\det(A) = \det(LU) = \det(L)\cdot\det(U) = \left(\prod L_{i,i}\right)\left(\prod U_{i,i}\right)$$

Отримаємо:

$$\Delta = \begin{vmatrix} 44.9 & 3.38 & -3 & 1 & -1.1 & 0.5 \\ 1.38 & 70.9 & -2.8 & 2.4 & -4.4 & -3.2 \\ 4.38 & 3.2 & 76.9 & 0.2 & 4.2 & -3.4 \\ 2.38 & 3.3 & -3.2 & 88.4 & -4.6 & 4.2 \\ 3.38 & -1.2 & -3.2 & 0.5 & 43.9 & 0.5 \\ 1.38 & 4.3 & 1.5 & -4.3 & -0.2 & 58.4 \end{vmatrix} = -26483964{,}518$$

Розрахуємо визначники матриць з підстановкою стовпця вільних членів на місце першого, другого, третього, чертвертого, п'ятого та шостого стовпців:

$$\Delta_{x_1} = \begin{vmatrix} -207.84 & 3.38 & -3 & 1 & -1.1 & 0.5 \\ -383.98 & 70.9 & -2.8 & 2.4 & -4.4 & -3.2 \\ 223.65 & 3.2 & 76.9 & 0.2 & 4.2 & -3.4 \\ -146.3 & 3.3 & -3.2 & 88.4 & -4.6 & 4.2 \\ 21.61 & -1.2 & -3.2 & 0.5 & 43.9 & 0.5 \\ 117.81 & 4.3 & 1.5 & -4.3 & -0.2 & 58.4 \end{vmatrix} = -225107839130.84$$

$$\Delta_{x_2} = \begin{vmatrix} 44.9 & -207.84 & -3 & 1 & -1.1 & 0.5 \\ 1.38 & -383.98 & -2.8 & 2.4 & -4.4 & -3.2 \\ 4.38 & 223.65 & 76.9 & 0.2 & 4.2 & -3.4 \\ 2.38 & -146.3 & -3.2 & 88.4 & -4.6 & 4.2 \\ 3.38 & 21.61 & -3.2 & 0.5 & 43.9 & 0.5 \\ 1.38 & 117.81 & 1.5 & -4.3 & -0.2 & 58.4 \end{vmatrix} = -281384798913.55$$

$$\Delta_{x_3} = \begin{vmatrix} 44.9 & 3.38 & -207.84 & 1 & -1.1 & 0.5 \\ 1.38 & 70.9 & -383.98 & 2.4 & -4.4 & -3.2 \\ 4.38 & 3.2 & 223.65 & 0.2 & 4.2 & -3.4 \\ 2.38 & 3.3 & -146.3 & 88.4 & -4.6 & 4.2 \\ 3.38 & -1.2 & 21.61 & 0.5 & 43.9 & 0.5 \\ 1.38 & 4.3 & 117.81 & -4.3 & -0.2 & 58.4 \end{vmatrix} = 191341663261.22$$

$$\Delta_{x_4} = \begin{vmatrix} 44.9 & 3.38 & -3 & -207.84 & -1.1 & 0.5 \\ 1.38 & 70.9 & -2.8 & -383.98 & -4.4 & -3.2 \\ 4.38 & 3.2 & 76.9 & 223.65 & 4.2 & -3.4 \\ 2.38 & 3.3 & -3.2 & -146.3 & -4.6 & 4.2 \\ 3.38 & -1.2 & -3.2 & 21.61 & 43.9 & 0.5 \\ 1.38 & 4.3 & 1.5 & 117.81 & -0.2 & 58.4 \end{vmatrix} = -73160047717.52$$

$$\Delta_{x_5} = \begin{vmatrix} 44.9 & 3.38 & -3 & 1 & -207.84 & 0.5 \\ 1.38 & 70.9 & -2.8 & 2.4 & -383.98 & -3.2 \\ 4.38 & 3.2 & 76.9 & 0.2 & 223.65 & -3.4 \\ 2.38 & 3.3 & -3.2 & 88.4 & -146.3 & 4.2 \\ 3.38 & -1.2 & -3.2 & 0.5 & 21.61 & 0.5 \\ 1.38 & 4.3 & 1.5 & -4.3 & 117.81 & 58.4 \end{vmatrix} = 50649263804.44$$

$$\Delta_{x_6} = \begin{vmatrix} 44.9 & 3.38 & -3 & 1 & -1.1 & -207.84 \\ 1.38 & 70.9 & -2.8 & 2.4 & -4.4 & -383.98 \\ 4.38 & 3.2 & 76.9 & 0.2 & 4.2 & 223.65 \\ 2.38 & 3.3 & -3.2 & 88.4 & -4.6 & -146.3 \\ 3.38 & -1.2 & -3.2 & 0.5 & 43.9 & 21.61 \\ 1.38 & 4.3 & 1.5 & -4.3 & -0.2 & 117.81 \end{vmatrix} = 129437007500.23$$

$$x_1 = \Delta_{x_1}/\Delta = -4.0$$

$$x_2 = \Delta_{x_2}/\Delta = -5.0$$

$$x_3 = \Delta_{x_3}/\Delta = 3.4$$

$$x_4 = \Delta_{x_4}/\Delta = -1.3$$

$$x_5 = \Delta_{x_5}/\Delta = 0.9$$

$$x_6 = \Delta_{x_6}/\Delta = 2.3$$

Отримаємо результати:

$$x_1 = -4, \quad x_2 = -5, \quad x_3 = 3.4, \quad x_4 = -1.3, \quad x_5 = 0.9, \quad x_6 = 2.3$$

Код програми на мові Python:

```
import numpy as np

def determinant(matrix):
    return round(np.linalg.det(matrix), 2)

def cramer_method(A, B):
    n = len(B)

    det_A = determinant(A)
    print("Крок 1: Визначник матриці коефіцієнтів (det(A)):")
    print(det_A)

    if det_A == 0:
        print("Визначник матриці коефіцієнтів дорівнює нулю, тому система не має єдиного розв'язку.")
        return

    #Обчислюємо кожну змінну за правилом Крамера
    solutions = []
    for i in range(n):
        # Створюємо копію матриці A
        Ai = np.copy(A)

        # Заміна i-го стовпця на вектор вільних членів B
        Ai[:, i] = B
```

```
        # Обчислюємо визначник зміненої матриці
        det_Ai = determinant(Ai)

        # Обчислюємо значення для i-ї змінної
        xi = det_Ai / det_A
        solutions.append(xi)

        # Вивід кроку
        print(f"\nКрок 2.{i+1}: Визначник матриці A з заміною {i+1}-го стовпця
(det(A_{i+1})): ")
        print(det_Ai)
        print(f"Розв'язок для x_{i+1}:")
        print(f"x_{i+1} = det(A_{i+1}) / det(A) = {det_Ai} / {det_A} = {xi}")

    # Виведення остаточного розв'язку
    print("\nОстаточні розв'язки:")
    for i, xi in enumerate(solutions):
        print(f"x_{i+1} = {xi}")

if __name__ == "__main__":
    A = np.array([[44.9, 3.38, -3, 1, -1.1, 0.5],
                  [1.38, 70.9, -2.8, 2.4, -4.4, -3.2],
                  [4.38, 3.2, 76.9, 0.2, 4.2, -3.4],
                  [2.38, 3.3, -3.2, 88.4, -4.6, 4.2],
                  [3.38, -1.2, -3.2, 0.5, 43.9, 0.5],
                  [1.38, 4.3, 1.5, -4.3, -0.2, 58.4]])

    # Вектор вільних членів B
    B = np.array([-207.84, -383.98, 223.64, -146.3, 21.61, 117.81])
```

```
cramer_method(A, B)
```

Метод обереної матриці

Метод оберненої матриці полягає в тому, що система лінійних рівнянь розв'язується за допомогою оберненої матриці коефіцієнтів. Формула має вигляд:

$$X = A^{-1} \cdot B$$

де - A^{-1} це обернена матриця коєфіцієнтів, B – вектор вільних членів, X – вектор невідомих.

Використаємо мову програмування Python та бібліотеку NumPy для обрахунків, спочатку обчислимо обернену матрицю за допомогою функції np.linalg.inv(), отримаємо:

$$A^{-1} = \begin{bmatrix} 0.0222 & -0.0011 & 0.0008 & -0.0002 & 0.0003 & -0.0002 \\ -0.0006 & 0.0141 & 0.0005 & -0.0003 & 0.0013 & 0.0008 \\ -0.0012 & -0.0006 & 0.0129 & 0.00004 & -0.0013 & 0.0007 \\ -0.0007 & -0.0004 & 0.0005 & 0.0113 & 0.0011 & -0.0008 \\ -0.0018 & 0.0004 & 0.0009 & -0.0001 & 0.0227 & -0.0001 \\ -0.0005 & -0.0010 & -0.0003 & 0.0009 & 0.0001 & 0.0170 \end{bmatrix}$$

Наступний крок:

$$A^{-1} \cdot B = \begin{bmatrix} 0.0222 & -0.0011 & 0.0008 & -0.0002 & 0.0003 & -0.0002 \\ -0.0006 & 0.0141 & 0.0005 & -0.0003 & 0.0013 & 0.0008 \\ -0.0012 & -0.0006 & 0.0129 & 0.00004 & -0.0013 & 0.0007 \\ -0.0007 & -0.0004 & 0.0005 & 0.0113 & 0.0011 & -0.0008 \\ -0.0018 & 0.0004 & 0.0009 & -0.0001 & 0.0227 & -0.0001 \\ -0.0005 & -0.0010 & -0.0003 & 0.0009 & 0.0001 & 0.0170 \end{bmatrix} \cdot \begin{bmatrix} -207.84 \\ -383.98 \\ 223.64 \\ -146.3 \\ 21.61 \\ 117.81 \end{bmatrix}$$

$$= \begin{bmatrix} -4 \\ -5 \\ 3.4 \\ -1.3 \\ 0.9 \\ 2.3 \end{bmatrix}$$

Отримуємо результат:

$$\boldsymbol{x_1 = -4, \quad x_2 = -5, \quad x_3 = 3.4, \quad x_4 = -1.3, \quad x_5 = 0.9, \quad x_6 = 2.3}$$

Код програми на мові Python:

```
import numpy as np
```

```
A = np.array([[44.9, 3.38, -3, 1, -1.1, 0.5],
          [1.38, 70.9, -2.8, 2.4, -4.4, -3.2],
          [4.38, 3.2, 76.9, 0.2, 4.2, -3.4],
          [2.38, 3.3, -3.2, 88.4, -4.6, 4.2],
          [3.38, -1.2, -3.2, 0.5, 43.9, 0.5],
          [1.38, 4.3, 1.5, -4.3, -0.2, 58.4]])


B = np.array([-207.84, -383.98, 223.64, -146.3, 21.61, 117.81])

A_inv = np.linalg.inv(A)

X = np.dot(A_inv, B)

print("Обернена матриця A:")
print(A_inv)

print("Розв'язки для змінних (x1, x2, x3, x4, x5, x6):")
print(X)
```

LU - декомпозиція

Метод розкладу квадратної матриці на добуток двох трикутних матриць: нижньої трикутної матриці L і верхньої трикутної матриці U.

$$A \cdot x = b$$

$$A = \begin{pmatrix} 44.9 & 3.38 & -3 & 1 & -1.1 & 0.5 \\ 1.38 & 70.9 & -2.8 & 2.4 & -4.4 & -3.2 \\ 4.38 & 3.2 & 76.9 & 0.2 & 4.2 & -3.4 \\ 2.38 & 3.3 & -3.2 & 88.4 & -4.6 & 4.2 \\ 3.38 & -1.2 & -3.2 & 0.5 & 43.9 & 0.5 \\ 1.38 & 4.3 & 1.5 & -4.3 & -0.2 & 58.4 \end{pmatrix} \quad b = \begin{pmatrix} -207.84 \\ -383.98 \\ 223.64 \\ -146.3 \\ 21.61 \\ 117.81 \end{pmatrix}$$

$$A = L \cdot U$$

$$L \cdot y = b$$

$$U \cdot x = y$$

*Де L – нижня трикутна матрица, U – верхня трикутна матриця**

Спочатку знайдемо нижню трикутну матрицю за допомогою бібліотеки NumPy та SciPy:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0.03 & 1 & 0 & 0 & 0 & 0 \\ 0.1 & 0.04 & 1 & 0 & 0 & 0 \\ 0.05 & 0.04 & -0.04 & 1 & 0 & 0 \\ 0.08 & -0.02 & -0.04 & 0.01 & 1 & 0 \\ 0.03 & 0.06 & 0.02 & -0.05 & -0.01 & 1 \end{pmatrix}$$

Тепер знайдемо U:

$$U = \begin{pmatrix} 44.9 & 3.38 & -3 & 1 & -1.1 & 0.5 \\ 0 & 70.8 & -2.71 & 2.37 & -4.37 & -3.22 \\ 0 & 0 & 77.3 & 0.01 & 4.48 & -3.32 \\ 0 & 0 & 0 & 88.24 & -4.18 & 4.19 \\ 0 & 0 & 0 & 0 & 44.09 & 0.24 \\ 0 & 0 & 0 & 0 & 0 & 58.86 \end{pmatrix}$$

Знайдемо *y* за допомогою функції np.linalg.solve(L,b) яка використовує чисельні методи та пряме підставлення:

$$y = \begin{pmatrix} -207.84 \\ -377.59 \\ 259.22 \\ -108.84 \\ 40.24 \\ 135.39 \end{pmatrix}$$

Тепер маємо можливість знайти x:

$$x = \begin{pmatrix} -4 \\ -5 \\ 3.4 \\ -1.3 \\ 0.9 \\ 2.3 \end{pmatrix}$$

Отримаємо результат:

$$x_1 = -4, \quad x_2 = -5, \quad x_3 = 3.4, \quad x_4 = -1.3, \quad x_5 = 0.9, \quad x_6 = 2.3$$

Код програми:

```
import numpy as np
from scipy.linalg import lu

A = np.array([[44.9, 3.38, -3, 1, -1.1, 0.5],
        [1.38, 70.9, -2.8, 2.4, -4.4, -3.2],
        [4.38, 3.2, 76.9, 0.2, 4.2, -3.4],
        [2.38, 3.3, -3.2, 88.4, -4.6, 4.2],
        [3.38, -1.2, -3.2, 0.5, 43.9, 0.5],
        [1.38, 4.3, 1.5, -4.3, -0.2, 58.4]])

b = np.array([-207.84, -383.98, 223.64, -146.3, 21.61, 117.81])

P, L, U = lu(A)

# (L * y = b)
y = np.linalg.solve(L, b)

# (U * x = y)
x = np.linalg.solve(U, y)

np.set_printoptions(precision=2, suppress=True)

print("P:\n", P, "\n")
print("L:\n", L, "\n")
print("U:\n", U, "\n")
print("y:\n", y, "\n")
```

```
print("x:\n", x, "\n")
```

Результати розрахунків (1):

Варіант №38			
	Крамера	Оберненої матриці	LU - декомпозиція
x_1	-4	-4	-4
x_2	-5	-5	-5
x_3	3.4	3.4	3.4
x_4	-1.3	-1.3	-1.3
x_5	0.9	0.9	0.9
x_6	2.3	2.3	2.3

Частина 2

3. Підготуватися до розв'язку системи лінійних рівнянь ітераційними методами:
    a. Якобі (також має назву «метод простої ітерації»);
    b. Зейделя (також має назву «метод Гаусса-Зейделя»).
4. Провести серію обчислювальних експериментів, що включають:
    a. розв'язок системи лінійних рівнянь з індивідуального завдання методами Якобі та Зейделя із двома-трьома різними значеннями точності ε (наприклад, ε = 0.1, ε =0.01, ε = 0.001, ε = 10−6);
    b. розв'язок системи лінійних рівнянь з індивідуального завдання методами Якобі та Зейделя із двома-трьома різними значеннями початкових наближень.
5. Зробити висновки про те, який з методів (Якобі чи Зейделя) збігається швидше, про те які фактори і в який спосіб впливають на збіжність ітераційних методів.

Якобі

Метод Якобі, також відомий як метод простої ітерації, є одним із числових методів розв'язання систем лінійних рівнянь. Цей метод заснований на ідеї послідовного наближення до розв'язку за допомогою ітераційного процесу.

Метод Якобі можна застосовувати до систем рівнянь, які мають вигляд:

$$A{\cdot}x = b$$

$$A = \begin{pmatrix} 44.9 & 3.38 & -3 & 1 & -1.1 & 0.5 \\ 1.38 & 70.9 & -2.8 & 2.4 & -4.4 & -3.2 \\ 4.38 & 3.2 & 76.9 & 0.2 & 4.2 & -3.4 \\ 2.38 & 3.3 & -3.2 & 88.4 & -4.6 & 4.2 \\ 3.38 & -1.2 & -3.2 & 0.5 & 43.9 & 0.5 \\ 1.38 & 4.3 & 1.5 & -4.3 & -0.2 & 58.4 \end{pmatrix} \quad b = \begin{pmatrix} -207.84 \\ -383.98 \\ 223.64 \\ -146.3 \\ 21.61 \\ 117.81 \end{pmatrix}$$

Розробимо програму на основі мови програмування Python та бібліотеки NumPy.

Перевірка працездатності розробленої програми:

Візьмемо рівняння з приклада до 2 частини лабораторної роботи та порівняємо результати.

Розв'язати методом простої ітерації з точністю $\varepsilon = 0{,}001$ систему лінійних рівнянь[1]:

$$\begin{cases} 0{,}80x_1 + 0{,}10x_2 + 0{,}10x_3 = 1840 \\ 0{,}20x_1 + 0{,}70x_2 + 0{,}10x_3 = 1860 \\ 0{,}05x_1 + 0{,}05x_2 + 0{,}90x_3 = 236. \end{cases}$$

Рис 2.1 – Завдання з прикладу

```
Ітерація     x1            x2            x3            Похибка
------------------------------------------------------------
0            2300.0000     2657.1        262.22        2657.1
1            1935.1        1962.5        -13.175       694.6
2            2056.3        2106.1        45.688        143.61
3            2031          2063.1        30.974        43.052
4            2038.2        2072.4        34.771        9.333
5            2036.6        2069.8        33.852        2.6056
6            2037          2070.4        34.088        0.60034
7            2036.9        2070.3        34.03         0.15961
8            2037          2070.3        34.045        0.038129
9            2037          2070.3        34.041        0.0098621
10           2037          2070.3        34.042        0.0024042
11           2037          2070.3        34.042        0.00061246

Точність досягнута через 11 ітерацій.

Розв’язок методом Якобі:
2037       2070.3     34.042
```

Рис 2.2 – Результати роботи програми

k	x_1	x_2	x_3	$\lvert x_1^{(k)} - x_1^{(k-1)} \rvert$	$\lvert x_2^{(k)} - x_2^{(k-1)} \rvert$	$\lvert x_3^{(k)} - x_3^{(k-1)} \rvert$	$\max \lvert x_i^{(k)} - x_i^{(k-1)} \rvert$	Закінч.обч
0	2300,0000	2657,1429	262,2222					
1	1935,0794	1962,5397	-13,1746	364,9206	694,6032	275,3968	694,6032	Продовж
2	2056,3294	2106,1451	45,6878	121,2500	143,6054	58,8624	143,6054	Продовж
3	2031,0209	2063,0933	30,9736	25,3085	43,0518	14,7142	43,0518	Продовж
4	2038,2416	2072,4264	34,7714	7,2207	9,3330	3,7978	9,3330	Продовж
5	2036,6003	2069,8208	33,8518	1,6414	2,6056	0,9197	2,6056	Продовж
6	2037,0409	2070,4211	34,0877	0,4407	0,6003	0,2359	0,6003	Продовж
7	2036,9364	2070,2615	34,0299	0,1045	0,1596	0,0578	0,1596	Продовж
8	2036,9636	2070,2996	34,0446	0,0272	0,0381	0,0147	0,0381	Продовж
9	2036,9570	2070,2898	34,0409	0,0066	0,0099	0,0036	0,0099	Закінч.

Рис 2.3 – Результат обрахунків з прикладу

Як видно обчислення дуже схожі та мають не вагому похибку. Тепер перевіримо на методі Гауса-Зейделя.

	k	x_1	x_2	x_3	$\lvert x_1^{(k)} - x_1^{(k-1)} \rvert$	$\lvert x_2^{(k)} - x_2^{(k-1)} \rvert$	$\lvert x_3^{(k)} - x_3^{(k-1)} \rvert$	$\max \lvert x_i^{(k)} - x_i^{(k-1)} \rvert$	Закінч.обч
14									
15	k	x_1	x_2	x_3	$\lvert x_1^{(k)} - x_1^{(k-1)} \rvert$	$\lvert x_2^{(k)} - x_2^{(k-1)} \rvert$	$\lvert x_3^{(k)} - x_3^{(k-1)} \rvert$	$\max \lvert x_i^{(k)} - x_i^{(k-1)} \rvert$	Закінч.обч
16	0	2300,0000	2657,1429	262,2222					
17	1	1935,0794	2066,8027	39,8954	364,921	590,340	222,327	590,3401	Продовж
18	2	2036,6627	2069,5399	34,0999	101,583	2,737	5,796	101,5834	Продовж
19	3	2037,0450	2070,2586	34,0387	0,382	0,719	0,061	0,7187	Продовж
20	4	2036,9628	2070,2908	34,0415	0,082	0,032	0,003	0,0822	Продовж
21	5	2036,9585	2070,2917	34,0417	0,004	0,001	0,000	0,0044	Продовж
22	6	2036,9583	2070,2917	34,0417	0,000	0,000	0,000	0,0001	Закінч.

Рис 2.4 – Результат обрахунків з прикладу

```
Ітерація      x1             x2             x3            Похибка
-----------------------------------------------------------
0            2300.0000     2000           23.333        2300.0000
1            2047.1        2068.9         33.555        252.92
2            2037.2        2070.3         34.029        9.8938
3            2037          2070.3         34.042        0.23004
4            2037          2070.3         34.042        0.0014506
5            2037          2070.3         34.042        0.00016175

Точність досягнута через 5 ітерацій.

Розв’язок методом Зейделя:
2037       2070.3     34.042
```

Рис 2.5 – Результат роботи програми (Метод Зейделя)

Результати збігаються.

Обрахунок індивідуального завдання:

Для наступних обрахунків використаємо 0 для початкового наближення (вектора x). Почнемо з точності ε = 0.1:

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
-----------------------------------------------------------------------------------------------
0           -4.629      -5.4158     2.9082      -1.655      0.49226     2.0173      5.4158
1           -4.0005     -5.0332     3.4638      -1.2931     0.90847     2.3306      0.62846
2           -3.9935     -4.9958     3.4023      -1.2974     0.90336     2.3014      0.061537

Точність досягнута через 2 ітерацій.

Розв'язок методом Якобі:
-3.9935    -4.9958    3.4023    -1.2974    0.90336    2.3014
```

Результат з точністю ε =0.01:

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
-----------------------------------------------------------------------------------------------
0           -4.629      -5.4158     2.9082      -1.655      0.49226     2.0173      5.4158
1           -4.0005     -5.0332     3.4638      -1.2931     0.90847     2.3306      0.62846
2           -3.9935     -4.9958     3.4023      -1.2974     0.90336     2.3014      0.061537
3           -4.0002     -4.9999     3.3993      -1.3001     0.89974     2.2997      0.0066333

Точність досягнута через 3 ітерацій.

Розв'язок методом Якобі:
-4.0002    -4.9999    3.3993    -1.3001    0.89974    2.2997
```

Результат з точністю ε = 0.001:

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
-----------------------------------------------------------------------------------------------
0           -4.629      -5.4158     2.9082      -1.655      0.49226     2.0173      5.4158
1           -4.0005     -5.0332     3.4638      -1.2931     0.90847     2.3306      0.62846
2           -3.9935     -4.9958     3.4023      -1.2974     0.90336     2.3014      0.061537
3           -4.0002     -4.9999     3.3993      -1.3001     0.89974     2.2997      0.0066333
4           -4.0001     -5          3.4         -1.3        0.89997     2.3         0.0006771

Точність досягнута через 4 ітерацій.

Розв'язок методом Якобі:
-4.0001    -5         3.4        -1.3       0.89997    2.3
```

Результатать з точністю ε = 10−6:

Ітерація	x1	x2	x3	x4	x5	x6	Похибка
0	-4.629	-5.4158	2.9082	-1.655	0.49226	2.0173	5.4158
1	-4.0005	-5.0332	3.4638	-1.2931	0.90847	2.3306	0.62846
2	-3.9935	-4.9958	3.4023	-1.2974	0.90336	2.3014	0.061537
3	-4.0002	-4.9999	3.3993	-1.3001	0.89974	2.2997	0.0066333
4	-4.0001	-5	3.4	-1.3	0.89997	2.3	0.0006771
5	-4	-5	3.4	-1.3	0.9	2.3	5.966e-05
6	-4	-5	3.4	-1.3	0.9	2.3	7.0887e-06
7	-4	-5	3.4	-1.3	0.9	2.3	4.9745e-07

Точність досягнута через 7 ітерацій.

Розв'язок методом Якобі:

-4 -5 3.4 -1.3 0.9 2.3

Замінемо початкове наближення на 100:

Результат з точністю ε = 0.1:

Ітерація	x1	x2	x3	x4	x5	x6	Похибка
0	-6.3661	3.9213	-8.2492	-4.0079	0.53781	-2.5717	108.25
1	-5.3442	-5.5647	2.975	-1.7784	0.56323	1.7976	11.224
2	-3.9779	-5.018	3.4975	-1.2518	0.96825	2.3479	1.3663
3	-3.9921	-4.9918	3.3978	-1.2951	0.90382	2.3021	0.099734

Точність досягнута через 3 ітерацій.

Розв'язок методом Якобі:

-3.9921 -4.9918 3.3978 -1.2951 0.90382 2.3021

Результат з точністю ε = 0.01:

Ітерація	x1	x2	x3	x4	x5	x6	Похибка
0	-6.3661	3.9213	-8.2492	-4.0079	0.53781	-2.5717	108.25
1	-5.3442	-5.5647	2.975	-1.7784	0.56323	1.7976	11.224
2	-3.9779	-5.018	3.4975	-1.2518	0.96825	2.3479	1.3663
3	-3.9921	-4.9918	3.3978	-1.2951	0.90382	2.3021	0.099734
4	-4.0008	-5.0001	3.3991	-1.3005	0.89937	2.2996	0.008738

Точність досягнута через 4 ітерацій.

Розв'язок методом Якобі:

-4.0008 -5.0001 3.3991 -1.3005 0.89937 2.2996

Результат з точністю ε = 0.001:

```
Ітерація     x1          x2          x3          x4          x5          x6          Похибка
-----------------------------------------------------------------------------------------------
0            -6.3661     3.9213      -8.2492     -4.0079     0.53781     -2.5717     108.25
1            -5.3442     -5.5647     2.975       -1.7784     0.56323     1.7976      11.224
2            -3.9779     -5.018      3.4975      -1.2518     0.96825     2.3479      1.3663
3            -3.9921     -4.9918     3.3978      -1.2951     0.90382     2.3021      0.099734
4            -4.0008     -5.0001     3.3991      -1.3005     0.89937     2.2996      0.008738
5            -4.0001     -5.0001     3.4001      -1.3        0.9         2.3         0.00099031

Точність досягнута через 5 ітерацій.

Розв’язок методом Якобі:
-4.0001    -5.0001    3.4001     -1.3       0.9        2.3
```

Результать з точністю ε = 10−6:

```
Ітерація     x1          x2          x3          x4          x5          x6          Похибка
-----------------------------------------------------------------------------------------------
0            -6.3661     3.9213      -8.2492     -4.0079     0.53781     -2.5717     108.25
1            -5.3442     -5.5647     2.975       -1.7784     0.56323     1.7976      11.224
2            -3.9779     -5.018      3.4975      -1.2518     0.96825     2.3479      1.3663
3            -3.9921     -4.9918     3.3978      -1.2951     0.90382     2.3021      0.099734
4            -4.0008     -5.0001     3.3991      -1.3005     0.89937     2.2996      0.008738
5            -4.0001     -5.0001     3.4001      -1.3        0.9         2.3         0.0009903
6            -4          -5          3.4         -1.3        0.90001     2.3         6.557e-05
7            -4          -5          3.4         -1.3        0.9         2.3         9.5074e-0
8            -4          -5          3.4         -1.3        0.9         2.3         1.093e-06
9            -4          -5          3.4         -1.3        0.9         2.3         1.1552e-0

Точність досягнута через 9 ітерацій.

Розв’язок методом Якобі:
-4         -5         3.4        -1.3       0.9        2.3
```

Метод Зейделя

Є вдосконаленням методу Якобі. Основна ідея полягає в тому, щоб на кожній ітерації використовувати вже оновлені значення змінних, що пришвидшує збіжність.

Обрахунов індивідуального завдання:

Для наступних обрахунків використаємо 0 для початкового наближення (вектора х). Результат з точністю ε = 0.1:

```
Ітерація     x1          x2          x3          x4          x5          x6          Похибка
-----------------------------------------------------------------------------------------------
0            -4.629      -5.3257     3.3935      -1.2087     0.9642      2.346       5.3257
1            -3.9769     -4.9977     3.3969      -1.2997     0.89753     2.2994      0.65206
2            -4.0004     -5.0003     3.4001      -1.3001     0.90004     2.3         0.023549

Точність досягнута через 2 ітерацій.

Розв’язок методом Зейделя:
-3.9769    -4.9977    3.3969     -1.2997    0.89753    2.2994
```

Результат з точністю ε =0.01:

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
------------------------------------------------------------------------------------------------
0           -4.629      -5.3257     3.3935      -1.2087     0.9642      2.346       5.3257
1           -3.9769     -4.9977     3.3969      -1.2997     0.89753     2.2994      0.65206
2           -4.0004     -5.0003     3.4001      -1.3001     0.90004     2.3         0.023549
3           -4          -5          3.4         -1.3        0.9         2.3         0.00047517

Точність досягнута через 3 ітерацій.

Розв'язок методом Зейделя:
-4.0004    -5.0003    3.4001     -1.3001    0.90004    2.3
```

Результат з точністю ε = 0.001:

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
------------------------------------------------------------------------------------------------
0           -4.629      -5.3257     3.3935      -1.2087     0.9642      2.346       5.3257
1           -3.9769     -4.9977     3.3969      -1.2997     0.89753     2.2994      0.65206
2           -4.0004     -5.0003     3.4001      -1.3001     0.90004     2.3         0.023549
3           -4          -5          3.4         -1.3        0.9         2.3         0.00047517

Точність досягнута через 3 ітерацій.

Розв'язок методом Зейделя:
-4.0004    -5.0003    3.4001     -1.3001    0.90004    2.3
```

Результать з точністю ε = 10−6:

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
------------------------------------------------------------------------------------------------
0           -4.629      -5.3257     3.3935      -1.2087     0.9642      2.346       5.3257
1           -3.9769     -4.9977     3.3969      -1.2997     0.89753     2.2994      0.65206
2           -4.0004     -5.0003     3.4001      -1.3001     0.90004     2.3         0.023549
3           -4          -5          3.4         -1.3        0.9         2.3         0.00047517
4           -4          -5          3.4         -1.3        0.9         2.3         3.633e-05
5           -4          -5          3.4         -1.3        0.9         2.3         1.1751e-06
6           -4          -5          3.4         -1.3        0.9         2.3         4.3127e-08

Точність досягнута через 6 ітерацій.

Розв'язок методом Зейделя:
-4         -5         3.4        -1.3       0.9        2.3
```

Замінемо початкове наближення на 100:

Результат з точністю ε = 0.1:

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
------------------------------------------------------------------------------------------------
0           -6.3661     5.9916      1.7211      -1.1925     0.14627     1.5951      106.37
1           -4.9526     -5.13       3.4694      -1.2727     0.98257     2.3326      11.122
2           -3.9845     -4.9919     3.3956      -1.2981     0.89832     2.2993      0.96809
3           -4.001      -5.0004     3.4001      -1.3        0.90008     2.3         0.016451

Точність досягнута через 3 ітерацій.

Розв'язок методом Зейделя:
-3.9845    -4.9919    3.3956     -1.2981    0.89832    2.2993
```

Результат з точністю ε = 0.01:

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
-------------------------------------------------------------------------------------------
0           -6.3661     5.9916      1.7211      -1.1925     0.14627     1.5951      106.37
1           -4.9526     -5.13       3.4694      -1.2727     0.98257     2.3326      11.122
2           -3.9845     -4.9919     3.3956      -1.2981     0.89832     2.2993      0.96809
3           -4.001      -5.0004     3.4001      -1.3        0.90008     2.3         0.016451
4           -4          -5          3.4         -1.3        0.9         2.3         0.0010133

Точність досягнута через 4 ітерацій.

Розв’язок методом Зейделя:
-4.001     -5.0004    3.4001     -1.3       0.90008    2.3
```

Результат з точністю ε = 0.001:

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
-------------------------------------------------------------------------------------------
0           -6.3661     5.9916      1.7211      -1.1925     0.14627     1.5951      106.37
1           -4.9526     -5.13       3.4694      -1.2727     0.98257     2.3326      11.122
2           -3.9845     -4.9919     3.3956      -1.2981     0.89832     2.2993      0.96809
3           -4.001      -5.0004     3.4001      -1.3        0.90008     2.3         0.016451
4           -4          -5          3.4         -1.3        0.9         2.3         0.0010133
5           -4          -5          3.4         -1.3        0.9         2.3         3.7998e-05

Точність досягнута через 5 ітерацій.

Розв’язок методом Зейделя:
-4         -5         3.4        -1.3       0.9        2.3
```

Результать з точністю ε = 10−6:

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
-------------------------------------------------------------------------------------------
0           -6.3661     5.9916      1.7211      -1.1925     0.14627     1.5951      106.37
1           -4.9526     -5.13       3.4694      -1.2727     0.98257     2.3326      11.122
2           -3.9845     -4.9919     3.3956      -1.2981     0.89832     2.2993      0.96809
3           -4.001      -5.0004     3.4001      -1.3        0.90008     2.3         0.016451
4           -4          -5          3.4         -1.3        0.9         2.3         0.0010133
5           -4          -5          3.4         -1.3        0.9         2.3         3.7998e-05
6           -4          -5          3.4         -1.3        0.9         2.3         1.3548e-06
7           -4          -5          3.4         -1.3        0.9         2.3         5.5024e-08

Точність досягнута через 7 ітерацій.

Розв’язок методом Зейделя:
-4         -5         3.4        -1.3       0.9        2.3
```

Код програми на мові Python:

```
import numpy as np


# функція для методу Якобі
def jacobi_method(A, b, eps=10**-6, max_iterations=100):
    # Ініціалізація змінних
```

```
    n = len(b)
    #x = np.zeros(n, dtype=np.double)
    x = np.full(n, 100, dtype=np.double)
    x_new = np.zeros_like(x, dtype=np.double)

    iteration = 0
    table_info = f'{"Ітерація":<12} ' + ''.join([f'x{i+1:<12}' for i in range(n)]) + 'Похибка'
    print(table_info)
    print('-' * len(table_info))

    # Початок циклу обчислень
    for iteration in range(max_iterations):
        for i in range(n):
            s = sum(A[i][j] * x[j] for j in range(n) if i != j)
            x_new[i] = (b[i] - s) / A[i][i]

            # Обчислення похибки
            error = np.linalg.norm(x_new - x, np.inf)

        # Проміжне виведення результатів
        res = f'{iteration:<12}' + ''.join([f'{format(value):<13}' for value in x_new]) + f'{format(error):<13}'
        print(res)

        # Перевірка на точність
        if error < eps:
            print("\nТочність досягнута через", iteration, "ітерацій.\n")
            break
```

```
        # Перезаписування значення
        x = x_new.copy()

    return x_new


def seidel_method(A, b, eps=10**-6, max_iterations=100):
    # Ініціалізація змінних
    n = len(b)
    x = np.full(n, 100, dtype=np.double)

    iteration = 0
    table_info = f'{"Ітерація":<12} ' + ''.join([f'x{i+1:<12}' for i in range(n)]) +
'Похибка'
    print(table_info)
    print('-' * len(table_info))

    for iteration in range(max_iterations):
        x_old = np.copy(x)
        for i in range(n):
            s1 = sum(A[i][j] * x_old[j] for j in range(i))
            s2 = sum(A[i][j] * x[j] for j in range(i + 1, n))
            x_old[i] = (b[i] - s1 - s2) / A[i][i]

        error = np.linalg.norm(x - x_old, np.inf)

        # Проміжне виведення результатів
        res = f'{iteration:<12}' + ''.join([f'{format(value):<13}' for value in x_old]) +
f'{format(error):<13}'
        print(res)
```

```
        # Перевірка на точність
        if error < eps:
            print("\nТочність досягнута через", iteration, "ітерацій.\n")
            break

        x = x_old.copy()

    return x

# Функція для форматування значень
def format(value):
    return f'{value:.4f}' if value.is_integer() else f'{value:.5g}'

def main():
    # Початкові дані
    A = np.array([
        [44.9, 3.38, -3, 1, -1.1, 0.5],
        [1.38, 70.9, -2.8, 2.4, -4.4, -3.2],
        [4.38, 3.2, 76.9, 0.2, 4.2, -3.4],
        [2.38, 3.3, -3.2, 88.4, -4.6, 4.2],
        [3.38, -1.2, -3.2, 0.5, 43.9, 0.5],
        [1.38, 4.3, 1.5, -4.3, -0.2, 58.4]
    ], dtype=float)

    b = np.array([-207.84, -383.98, 223.64, -146.3, 21.61, 117.81], dtype=float)

    # Виведення матриці та вектора
    print("Матриця A:")
    for row in A: print(' '.join([f'{format(value):<10}' for value in row]))
    print("\nВектор b:")
```

```
    print(' '.join([f'{format(value):<5}' for value in b]))
    print()

    solution_jacobi = jacobi_method(A, b)
    print(f'Розв’язок методом Якобі:')
    print(' '.join([f'{format(value):<10}' for value in solution_jacobi]))

    print("\n")

    print("Матриця A:")
    for row in A: print(' '.join([f'{format(value):<10}' for value in row]))
    print("\nВектор b:")
    print(' '.join([f'{format(value):<5}' for value in b]))
    print()
    solution_seidel = seidel_method(A, b)
    print(f'Розв’язок методом Зейделя:')
    print(' '.join([f'{format(value):<10}' for value in solution_seidel]))


if __name__ == '__main__':
    main()
```

Результати розрахунків (2)

Варіант №38									
	Поч. наближ	Якобі				Зейделя			
		$\varepsilon = 0.1$	ε = 0.01	ε = 0.001	ε = 10^{-6}	$\varepsilon = 0.1$	ε = 0.01	ε = 0.001	ε =10^{-6}
x_1	0	-3.99	-4.00	-4.00	-4.00	-3.97	-4.00	-4.00	-4.00
x_2	0	-4.99	-4.99	-5.00	-5.00	-4.99	-5.00	-5.00	-5.00
x_3	0	3.40	3.39	3.40	3.40	3.39	3.40	3.40	3.40
x_4	0	-1.29	-1.30	-1.3	-1.30	-1.29	-1.30	-1.30	-1.30
x_5	0	0.90	0.89	0.90	0.90	0.89	0.90	0.90	0.90
x_6	0	2.30	2.29	2.30	2.30	2.29	2.30	2.30	2.30
Кільк. ітерацій	-	2	3	5	7	2	3	5	6

Результати розрахунків (3):

Варіант №38									
	Поч. наближ	Якобі				Зейделя			
		$\varepsilon = 0.1$	ε = 0.01	ε = 0.001	ε = 10^{-6}	$\varepsilon = 0.1$	ε = 0.01	ε = 0.001	ε = 10^{-6}
x_1	100	-3.99	-4.00	-4.00	-4.00	-3.98	-4.00	-4.00	-4.00
x_2	100	-4.99	-5.00	-5.00	-5.00	-4.99	-5.00	-5.00	-5.00
x_3	100	3.39	3.39	3.40	3.40	3.39	3.40	3.40	3.40
x_4	100	-1.29	-1.30	-1.3	-1.30	-1.29	-1.30	-1.30	-1.30
x_5	100	0.90	0.89	0.89	0.90	0.89	0.90	0.90	0.90
x_6	100	2.30	2.29	2.30	2.30	2.29	2.30	2.30	2.30
Кільк. ітерацій	-	3	4	4	9	3	4	3	7

Висновки:

Загалом швидкості двох методів схожі, але метод Зейделя виявляється швидшим за метод Якобі для розв’язання цієї системи рівнянь. При збільшенні точності ε =10^{-6} метод Зейделя потребує 6 ітерацій, тоді як Якобі — 7 ітерацій (для першого початкового наближення x=0). При початковому значенні x=100 методи Якобі та Зейделя потребують трохи більше ітерацій. Це показує, що початкове наближення може впливати на кількість ітерацій, але метод Зейделя все одно зберігає свою перевагу в швидкості збіжності. Зі зменшенням ε (від 0.1 до 10^{-6}) обидва методи потребують більше ітерацій для досягнення збіжності. Метод Зейделя сходиться швидше, але обидва методи чутливі до початкового наближення та вимагаються більше ітерацій при підвищенні точності.

Частина 3:

6. Підготуватися до розв’язку системи лінійних рівнянь методом верхньої релаксації (SOR-методом, $\omega>1$). В рамках підготовки рекомендовано написати програму однією з мов програмування, як реалізують розв’язок системи лінійних алгебраїчних рівнянь довільної розмірності.

Також, рекомендовано передбачити (опціональне) проміжне виведення значень змінних та інших характеристик на кожній ітерації (у вигляді текстової таблиці).

Працездатність розробленої програми (таблицю ходу розрахунку, відповідь) перевірити на тестовому прикладі з літератури.

Примітка. Оскільки метод Зейделя (див. п. 3) може бути розглянутий як частковий випадок методу релаксації, припускається підготувати одну “універсальну” програму для розв’язання системи лінійних алгебраїчних рівнянь методами Зейделя (із подальшим введенням користувачем значення $\omega=1$) та методом верхньої релаксації (із введенням $\omega>1$).

7. Провести серію експериментів із варіацією двох-трьох різних значень ε, двох-трьох різних значень початкових наближень і двох-трьох різних значень релаксаційного параметру ω.

8. Зробити висновки оптимального значення параметру ω для випадку індивідуального завдання свого варіанту.

9. Представити результати викладачеві.

10. Оформити звіт з лабораторної роботи відповідно до вимог.

Метод верхньої релаксації (SOR, $\omega > 1$)

Метод SOR вводить додатковий параметр релаксації (ω), який дозволяє прискорити збіжність ітераційного процесу шляхом модифікації оновлення кожної невідомої.

Індивідуальне завдання:

$$A = \begin{pmatrix} 44.9 & 3.38 & -3 & 1 & -1.1 & 0.5 \\ 1.38 & 70.9 & -2.8 & 2.4 & -4.4 & -3.2 \\ 4.38 & 3.2 & 76.9 & 0.2 & 4.2 & -3.4 \\ 2.38 & 3.3 & -3.2 & 88.4 & -4.6 & 4.2 \\ 3.38 & -1.2 & -3.2 & 0.5 & 43.9 & 0.5 \\ 1.38 & 4.3 & 1.5 & -4.3 & -0.2 & 58.4 \end{pmatrix} \quad b = \begin{pmatrix} -207.84 \\ -383.98 \\ 223.64 \\ -146.3 \\ 21.61 \\ 117.81 \end{pmatrix}$$

- Якщо ω = 1, метод SOR стає методом Гаусса–Зейделя.
- Якщо ω > 1, використовується верхня релаксація.
- Якщо 0 < ω < 1, застосовується недорелаксація.

Правильний вибір ω може значно вплинути на швидкість збіжності методу SOR. Зазвичай оптимальний параметр релаксації знаходиться в інтервалі від 1 до 2 для верхньої релаксації. Якщо ω занадто велике, метод може не сходитися.

Напишемо програму для розрахунку за допомогою мови Python та вище згаданої бібліотеки NumPy.

Код програми:

```
import numpy as np



def sor_method(A, b, omega=1.5, eps=10**-6, max_iterations=100):
    # Ініціалізація змінних
    n = len(b)
    # x = np.array([b[i] / A[i][i] for i in range(n)], dtype=np.double)
    x = np.ones_like(b, dtype=np.double)

    iteration = 0
    table_info = (
        f'{"Ітерація":<12} ' + "".join([f"x{i+1:<12}" for i in range(n)]) + "Похибка"
    )
```

```
print(table_info)
print("-" * len(table_info))

# Початок циклу обчислень
for iteration in range(max_iterations):
    x_old = np.copy(x)
    for i in range(n):
        s1 = sum(A[i][j] * x_old[j] for j in range(i))
        s2 = sum(A[i][j] * x[j] for j in range(i + 1, n))
        # Метод SOR використовує параметр ω для оновлення значення x[i]
        x_old[i] = (1 - omega) * x[i] + (omega / A[i][i]) * (b[i] - s1 - s2)

    error = np.linalg.norm(x - x_old, np.inf)

    x = x_old.copy()
    # Проміжне виведення результатів
    print(
        f"{iteration + 1:<12}"
        + "".join([f"{value:<13.3f}" for value in x])
        + f"{error:<13.6e}"
    )

    # Перевірка на точність
    if error < eps:
        print("\nТочність досягнута через", iteration + 1, "ітерацій.\n")
        break

return x
```

```
def main():

    # Початкові дані

    A = np.array(
        [
            [44.9, 3.38, -3, 1, -1.1, 0.5],
            [1.38, 70.9, -2.8, 2.4, -4.4, -3.2],
            [4.38, 3.2, 76.9, 0.2, 4.2, -3.4],
            [2.38, 3.3, -3.2, 88.4, -4.6, 4.2],
            [3.38, -1.2, -3.2, 0.5, 43.9, 0.5],
            [1.38, 4.3, 1.5, -4.3, -0.2, 58.4],
        ],
        dtype=float,
    )

    b = np.array([-207.84, -383.98, 223.64, -146.3, 21.61, 117.81], dtype=float)

    # Виведення матриці та вектора
    print("Матриця A:")
    for row in A:
        print(" ".join([f"{format(value):<10}" for value in row]))
    print("\nВектор b:")
    print(" ".join([f"{format(value):<5}" for value in b]))
    print()

    # Виведення результатів
    solution = sor_method(A, b)
    print(f"Розв’язок методом SOR:")
    print(" ".join([f"{format(value):<10.4}" for value in solution]))
```

```python
if __name__ == "__main__":
    main()
```

Перевірка працездатності розробленої програми:

Візьмемо завдання з прикладу до 3 частини лабораторної роботи, де

$$A = \begin{matrix} 0.80 & 0.10 & 0.10 \\ 0.20 & 0.70 & 0.10 \\ 0.05 & 0.05 & 0.90 \end{matrix} \quad b = \begin{matrix} 1840 \\ 1860 \\ 236 \end{matrix}$$

Початкове наближення (x,y,z)=(2300,2657.1,262.2), а значення релаксації - w=1.1.

Отримані результати в прикладі:

Перша ітерація:

x_1=(1-1.1)·2300+1.1·10.8[1840-0.1(2657.1)-0.1(262.2)] = (-0.1)·2300+1.1·10.8[1548.07]=-230+2128.596 = 1898.596
y_1=(1-1.1)·2657.1+1.1·10.7[1860-0.2(1898.596)-0.1(262.2)]=(-0.1)·2657.1+1.1·10.7[1454.061]=-265.71+2284.953=2019.243
z_1=(1-1.1)·262.2+1.1·10.9[236-0.05(1898.596)-0.05(2019.243)]=(-0.1)·262.2+1.1·10.9[40.108]=-26.22+49.021=22.801

Друга ітерація:

x_2=(1-1.1)·1898.596+1.1·10.8[1840-0.1(2019.243)-0.1(22.801)]=(-0.1)·1898.596+1.1·10.8[1635.796]=-189.86+2249.219=2059.359
y_2=(1-1.1)·2019.243+1.1·10.7[1860-0.2(2059.359)-0.1(22.801)]=(-0.1)·2019.243+1.1·10.7[1445.848]=-201.924+2272.047=2070.123
z_2=(1-1.1)·22.801+1.1·10.9[236-0.05(2059.359)-0.05(2070.123)]=(-0.1)·22.801+1.1·10.9[29.526]=-2.28+36.087=33.807

Третя ітерація:

x_3=(1-1.1)·2059.359+1.1·10.8[1840-0.1(2070.123)-0.1(33.807)]=(-0.1)·2059.359+1.1·10.8[1629.607]=-205.936+2240.71=2034.774
y_3=(1-1.1)·2070.123+1.1·10.7[1860-0.2(2034.774)-0.1(33.807)]=(-0.1)·2070.123+1.1·10.7[1449.665]=-207.012+2278.044=2071.032
z_3=(1-1.1)·33.807+1.1·10.9[236-0.05(2034.774)-0.05(2071.032)]=(-0.1)·33.807+1.1·10.9[30.71]=-3.381+37.534=34.153

Порівняємо з результатами розробленої програми:

Ітерація	x1	x2	x3	Похибка
1	1898.596	2019.243	22.801	6.378574e+02
2	2059.359	2070.123	33.807	1.607631e+02
3	2034.774	2071.032	34.153	2.458566e+01

Як видно в результаті роботи програми, результат на всіх ітераціях програми збігається з прикладом.

Проведемо серію експериментів із варіацією двох різних значень ε, двох різних значень початкових наближень і двох різних значень релаксаційного параметру ω.

Спочатку початкові наближення будуть у вигляді всіх 0:

$$\varepsilon = 0.1 \ \omega = 1.1$$

Ітерація	x1	x2	x3	x4	x5	x6	Похибка
1	-5.092	-5.848	3.786	-1.279	1.116	2.619	5.848358e+00
2	-3.791	-4.873	3.345	-1.320	0.856	2.252	1.301041e+00
3	-4.036	-5.019	3.409	-1.296	0.908	2.307	2.447706e-01
4	-3.994	-4.997	3.398	-1.301	0.899	2.299	4.138688e-02

```
Точність досягнута через 4 ітерацій.

Розв'язок методом SOR:
-3.9     -4.9     3.39     -1.3     0.89     2.29
```

$$\varepsilon = 10^{-6} \ \omega = 1.1$$

Ітерація	x1	x2	x3	x4	x5	x6	Похибка
1	-5.092	-5.848	3.786	-1.279	1.116	2.619	5.848358e+00
2	-3.791	-4.873	3.345	-1.320	0.856	2.252	1.301041e+00
3	-4.036	-5.019	3.409	-1.296	0.908	2.307	2.447706e-01
4	-3.994	-4.997	3.398	-1.301	0.899	2.299	4.138688e-02
5	-4.001	-5.000	3.400	-1.300	0.900	2.300	6.751769e-03
6	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.096678e-03
7	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.795698e-04
8	-4.000	-5.000	3.400	-1.300	0.900	2.300	2.971362e-05
9	-4.000	-5.000	3.400	-1.300	0.900	2.300	4.959732e-06
10	-4.000	-5.000	3.400	-1.300	0.900	2.300	8.327490e-07

```
Точність досягнута через 10 ітерацій.

Розв'язок методом SOR:
-3.9     -4.9     3.39     -1.3     0.89     2.29
```

$$\varepsilon = 0.1 \omega = 1.5$$

Ітерація	x1	x2	x3	x4	x5	x6	Похибка
1	-6.943	-7.921	5.450	-1.463	1.836	3.785	7.920975e+00
2	-1.978	-3.281	2.117	-1.499	0.107	1.319	4.965463e+00
3	-5.340	-6.026	4.221	-1.036	1.511	2.952	3.361814e+00
4	-3.129	-4.376	2.868	-1.529	0.454	1.867	2.210587e+00
5	-4.561	-5.386	3.747	-1.126	1.214	2.586	1.431439e+00
6	-3.640	-4.757	3.173	-1.423	0.684	2.111	9.202512e-01
7	-4.231	-5.155	3.548	-1.216	1.046	2.424	5.902485e-01
8	-3.852	-4.900	3.303	-1.356	0.802	2.219	3.787889e-01
9	-4.095	-5.065	3.464	-1.263	0.965	2.353	2.435850e-01
10	-3.938	-4.958	3.358	-1.324	0.857	2.265	1.570624e-01
11	-4.040	-5.027	3.427	-1.284	0.928	2.323	1.015553e-01
12	-3.974	-4.982	3.382	-1.310	0.881	2.285	6.583359e-02

Точність досягнута через 12 ітерацій.

Розв'язок методом SOR:

```
-3.9    -4.9    3.38    -1.3    0.88    2.28
```

$$\varepsilon = 10^{-6} \omega = 1.5$$

Ітерація	x1	x2	x3	x4	x5	x6	Похибка
1	-6.943	-7.921	5.450	-1.463	1.836	3.785	7.920975e+00
2	-1.978	-3.281	2.117	-1.499	0.107	1.319	4.965463e+00
3	-5.340	-6.026	4.221	-1.036	1.511	2.952	3.361814e+00
4	-3.129	-4.376	2.868	-1.529	0.454	1.867	2.210587e+00
5	-4.561	-5.386	3.747	-1.126	1.214	2.586	1.431439e+00
6	-3.640	-4.757	3.173	-1.423	0.684	2.111	9.202512e-01
7	-4.231	-5.155	3.548	-1.216	1.046	2.424	5.902485e-01
8	-3.852	-4.900	3.303	-1.356	0.802	2.219	3.787889e-01
9	-4.095	-5.065	3.464	-1.263	0.965	2.353	2.435850e-01
10	-3.938	-4.958	3.358	-1.324	0.857	2.265	1.570624e-01
11	-4.040	-5.027	3.427	-1.284	0.928	2.323	1.015553e-01
12	-3.974	-4.982	3.382	-1.310	0.881	2.285	6.583359e-02
13	-4.017	-5.012	3.412	-1.293	0.912	2.310	4.277088e-02
14	-3.989	-4.992	3.392	-1.304	0.892	2.294	2.783726e-02
15	-4.007	-5.005	3.405	-1.297	0.905	2.304	1.814312e-02
16	-3.995	-4.997	3.397	-1.302	0.897	2.297	1.183739e-02
17	-4.003	-5.002	3.402	-1.299	0.902	2.302	7.729209e-03
18	-3.998	-4.999	3.399	-1.301	0.899	2.299	5.049554e-03
19	-4.001	-5.001	3.401	-1.299	0.901	2.301	3.300171e-03
20	-3.999	-4.999	3.399	-1.300	0.899	2.300	2.157406e-03
21	-4.001	-5.000	3.400	-1.300	0.900	2.300	1.410590e-03
22	-4.000	-5.000	3.400	-1.300	0.900	2.300	9.223947e-04
23	-4.000	-5.000	3.400	-1.300	0.900	2.300	6.032015e-04
24	-4.000	-5.000	3.400	-1.300	0.900	2.300	3.944809e-04
25	-4.000	-5.000	3.400	-1.300	0.900	2.300	2.579886e-04
26	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.687258e-04
27	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.103486e-04
28	-4.000	-5.000	3.400	-1.300	0.900	2.300	7.216976e-05
29	-4.000	-5.000	3.400	-1.300	0.900	2.300	4.720044e-05
30	-4.000	-5.000	3.400	-1.300	0.900	2.300	3.087019e-05
31	-4.000	-5.000	3.400	-1.300	0.900	2.300	2.018992e-05
32	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.320482e-05
33	-4.000	-5.000	3.400	-1.300	0.900	2.300	8.636401e-06
34	-4.000	-5.000	3.400	-1.300	0.900	2.300	5.648532e-06
35	-4.000	-5.000	3.400	-1.300	0.900	2.300	3.694373e-06
36	-4.000	-5.000	3.400	-1.300	0.900	2.300	2.416287e-06
37	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.580369e-06
38	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.033643e-06
39	-4.000	-5.000	3.400	-1.300	0.900	2.300	6.760596e-07

Точність досягнута через 39 ітерацій.

Розв'язок методом SOR:

```
-4.0    -5.0    3.40    -1.2    0.90    2.30
```

Поміняємо початкові наближення на 1:

$$\varepsilon = 0.1 \ \omega = 1.1$$

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
----------------------------------------------------------------------------------------------
1           -5.211      -5.822      3.678       -1.376      1.007       2.514       6.821689e+00
2           -3.788      -4.890      3.358       -1.310      0.869       2.264       1.422622e+00
3           -4.034      -5.016      3.407       -1.297      0.906       2.306       2.452207e-01
4           -3.995      -4.998      3.399       -1.301      0.899       2.299       3.876901e-02

Точність досягнута через 4 ітерацій.

Розв'язок методом SOR:
-3.9      -4.9      3.39      -1.3      0.89      2.29
```

$$\varepsilon = 10^{-6} \ \omega = 1.1$$

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
----------------------------------------------------------------------------------------------
1           -5.211      -5.822      3.678       -1.376      1.007       2.514       6.821689e+00
2           -3.788      -4.890      3.358       -1.310      0.869       2.264       1.422622e+00
3           -4.034      -5.016      3.407       -1.297      0.906       2.306       2.452207e-01
4           -3.995      -4.998      3.399       -1.301      0.899       2.299       3.876901e-02
5           -4.001      -5.000      3.400       -1.300      0.900       2.300       6.032712e-03
6           -4.000      -5.000      3.400       -1.300      0.900       2.300       9.518570e-04
7           -4.000      -5.000      3.400       -1.300      0.900       2.300       1.535614e-04
8           -4.000      -5.000      3.400       -1.300      0.900       2.300       2.526349e-05
9           -4.000      -5.000      3.400       -1.300      0.900       2.300       4.212322e-06
10          -4.000      -5.000      3.400       -1.300      0.900       2.300       7.078045e-07

Точність досягнута через 10 ітерацій.

Розв'язок методом SOR:
-3.9      -4.9      3.39      -1.3      0.89      2.29
```

$$\varepsilon = 0.1 \omega = 1.5$$

```
Ітерація    x1          x2          x3          x4          x5          x6          Похибка
----------------------------------------------------------------------------------------------
1           -7.469      -8.236      4.995       -1.942      1.326       3.300       9.236363e+00
2           -1.720      -3.214      2.330       -1.267      0.362       1.564       5.749898e+00
3           -5.458      -6.015      4.118       -1.151      1.384       2.823       3.738055e+00
4           -3.081      -4.404      2.920       -1.471      0.518       1.938       2.377141e+00
5           -4.578      -5.361      3.720       -1.158      1.182       2.546       1.497008e+00
6           -3.637      -4.775      3.188       -1.405      0.701       2.135       9.406076e-01
7           -4.229      -5.143      3.540       -1.226      1.037       2.410       5.923290e-01
8           -3.854      -4.908      3.307       -1.350      0.807       2.227       3.747589e-01
9           -4.093      -5.060      3.461       -1.267      0.962       2.348       2.384621e-01
10          -3.940      -4.961      3.360       -1.322      0.859       2.269       1.526146e-01
11          -4.039      -5.026      3.426       -1.286      0.927       2.321       9.818917e-02

Точність досягнута через 11 ітерацій.

Розв'язок методом SOR:
-4.0      -5.0      3.42      -1.2      0.92      2.32
```

$$\varepsilon = 10^{-6} \omega = 1.5$$

Ітерація	x1	x2	x3	x4	x5	x6	Похибка
1	-7.469	-8.236	4.995	-1.942	1.326	3.300	9.236363e+00
2	-1.720	-3.214	2.330	-1.267	0.362	1.564	5.749898e+00
3	-5.458	-6.015	4.118	-1.151	1.384	2.823	3.738055e+00
4	-3.081	-4.404	2.920	-1.471	0.518	1.938	2.377141e+00
5	-4.578	-5.361	3.720	-1.158	1.182	2.546	1.497008e+00
6	-3.637	-4.775	3.188	-1.405	0.701	2.135	9.406076e-01
7	-4.229	-5.143	3.540	-1.226	1.037	2.410	5.923290e-01
8	-3.854	-4.908	3.307	-1.350	0.807	2.227	3.747589e-01
9	-4.093	-5.060	3.461	-1.267	0.962	2.348	2.384621e-01
10	-3.940	-4.961	3.360	-1.322	0.859	2.269	1.526146e-01
11	-4.039	-5.026	3.426	-1.286	0.927	2.321	9.818917e-02
12	-3.975	-4.983	3.383	-1.309	0.882	2.286	6.345708e-02
13	-4.016	-5.011	3.411	-1.294	0.912	2.309	4.115922e-02
14	-3.989	-4.993	3.393	-1.304	0.892	2.294	2.677101e-02
15	-4.007	-5.005	3.405	-1.297	0.905	2.304	1.744856e-02
16	-3.995	-4.997	3.397	-1.302	0.897	2.298	1.138925e-02
17	-4.003	-5.002	3.402	-1.299	0.902	2.302	7.441689e-03
18	-3.998	-4.999	3.399	-1.301	0.899	2.299	4.865635e-03
19	-4.001	-5.001	3.401	-1.300	0.901	2.301	3.182665e-03
20	-3.999	-4.999	3.399	-1.300	0.899	2.300	2.082334e-03
21	-4.001	-5.000	3.400	-1.300	0.900	2.300	1.362595e-03
22	-4.000	-5.000	3.400	-1.300	0.900	2.300	8.916768e-04
23	-4.000	-5.000	3.400	-1.300	0.900	2.300	5.835149e-04
24	-4.000	-5.000	3.400	-1.300	0.900	2.300	3.818462e-04
25	-4.000	-5.000	3.400	-1.300	0.900	2.300	2.498683e-04
26	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.634998e-04
27	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.069810e-04
28	-4.000	-5.000	3.400	-1.300	0.900	2.300	6.999713e-05
29	-4.000	-5.000	3.400	-1.300	0.900	2.300	4.579724e-05
30	-4.000	-5.000	3.400	-1.300	0.900	2.300	2.996301e-05
31	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.960290e-05
32	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.282464e-05
33	-4.000	-5.000	3.400	-1.300	0.900	2.300	8.389997e-06
34	-4.000	-5.000	3.400	-1.300	0.900	2.300	5.488717e-06
35	-4.000	-5.000	3.400	-1.300	0.900	2.300	3.590652e-06
36	-4.000	-5.000	3.400	-1.300	0.900	2.300	2.348930e-06
37	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.536603e-06
38	-4.000	-5.000	3.400	-1.300	0.900	2.300	1.005191e-06
39	-4.000	-5.000	3.400	-1.300	0.900	2.300	6.575542e-07

Точність досягнута через 39 ітерацій.

Розв'язок методом SOR:

-4.0 -5.0 3.40 -1.2 0.90 2.30

Результати розрахунків (4)

Варіант №38					
Метод верхньої релаксації					
	Поч. наближ.	$\varepsilon = 0.1$ $\omega = 1.1$	$\varepsilon = 10^{-6}$ $\omega = 1.1$	$\varepsilon = 0.1$ $\omega = 1.5$	$\varepsilon = 10^{-6}$ $\omega = 1.5$
x_1	0	-3.9	-3.9	-3.9	-4.0
x_2	0	-4.9	-4.9	-4.9	-5.0
x_3	0	3.39	3.39	3.38	3.40
x_4	0	-1.3	-1.3	-1.3	-1.2
x_5	0	0.89	0.89	0.88	0.90
x_6	0	2.29	2.29	2.28	2.30
Кільк. ітерацій	-	4	10	12	39

Результати розрахунків (5)

Варіант №38					
Метод верхньої релаксації					
	Поч. наближ.	$\varepsilon = 0.1$ $\omega = 1.1$	$\varepsilon = 10^{-6}$ $\omega = 1.1$	$\varepsilon = 0.1$ $\omega = 1.5$	$\varepsilon = 10^{-6}$ $\omega = 1.5$
x_1	1	-3.9	-3.9	-4.0	-4.0
x_2	1	-4.9	-4.9	-5.0	-5.0
x_3	1	3.39	3.39	3.42	3.40
x_4	1	-1.3	-1.3	-1.2	-1.2
x_5	1	0.89	0.89	0.92	0.90
x_6	1	2.29	2.29	2.32	2.30
Кільк. ітерацій	-	4	10	11	39

Висновки:

Зменшення точності ε з 0.1 до 10^{-6} збільшує кількість ітерацій для обох значень ω. Це очікувано, оскільки для більш високої точності потрібні додаткові ітерації для досягнення збіжності. Для ω=1.1, кількість ітерацій менша, ніж для

$\omega=1.5$, за однакових значень точності ε. Це свідчить про те, що при $\omega=1.1$ метод SOR має кращу збіжність для цієї конкретної задачі. Початкові значення (0 або 1) суттєво не впливають на кількість ітерацій — результати майже ідентичні. Це свідчить про те, що для даної задачі метод SOR з обраними параметрами сходиться досить стабільно, незалежно від початкових умов або що різниця між обраними параметрами наближення не настільки велика щоб побачити різницю в ефективності. На основі результатів, $\omega=1.1$ є більш оптимальним вибором для даної задачі, оскільки потребує меншої кількості ітерацій у порівнянні з $\omega=1.5$. При надто високому значенні ω метод може стати менш ефективним і навіть втратити збіжність як це стало коли кількість ітерацій набула 39.