НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра штучного інтелекту

ЛАБОРАТОРНА РОБОТА №2

з дисципліни «Проектування та аналіз обчислювальних алгоритмів»

Варіант № 38

Виконав:

Студент II курсу

Групи КІ-32

Присяжнюк Владислав

Прийняв:

Тимошенко Ю. О.

Київ 2024

Білет 57

1. Розв’яжіть систему лінійних алгебраїчних рівнянь (СЛАР) певного розміру згідно вимогам по лабораторній роботі No2 з заданою некоректною матрицею A певного розміру, використавши задані два класичні алгоритми розв’язку СЛАР. Використайте наступні класичні алгоритми:

- Gauss-Jordan Elimination method
- LU Decomposition
-

$$A = \begin{matrix} 1 & 0.99 & 0.98 & 0.97 & 0.96 \\ 0.99 & 1 & 0.99 & 0.98 & 0.97 \\ 0.98 & 0.99 & 1 & 0.99 & 0.98 \\ 0.97 & 0.98 & 0.99 & 1 & 0.99 \\ 0.96 & 0.97 & 0.98 & 0.99 & 1 \end{matrix}$$

2. Здійсніть перевірку розширеного правила Крамера згідно додатку 2.

Оскільки вектор b нам не наданий то обрахуємо його самостійно, для цього заповнимо вектор x одиницями, та отримаємо що вектор b стає рівним:

$$b = \begin{matrix} 4.9 \\ 4.93 \\ 4.94 \\ 4.93 \\ 4.9 \end{matrix}$$

Теоретичні відомсті:

Метод Гауса-Жордана - це модифікація методу Гауса, яка дозволяє знайти як розв’язок системи лінійних рівнянь (СЛАР), так і обернену матрицю.

Основні кроки алгоритму:

- Приведення матриці до діагональної форми за допомогою елементарних рядкових операцій.
- Нормалізація кожного рядка шляхом ділення на діагональний елемент.
- Використання цих нормалізованих рядків для обнулення інших елементів у стовпці.

Часова складність: $O(n^3)$, де n — розмір матриці.

Метод LU-розкладу - LU-розклад розбиває матрицю A на дві матриці: нижню трикутну L і верхню трикутну U, так що A=L·U.

Основні етапи:

- Елементи матриці A розкладаються на елементи L (нижньої трикутної матриці) і U (верхньої трикутної матриці).
- L має одиниці на діагоналі.
- Спочатку розв'язується проміжна система L·y=b (методом прямої підстановки).
- Потім розв'язується система U·x=y (методом зворотної підстановки).

Часова складність: $O(n^3)$ для розкладання, але розв'язання кожної системи із фіксованою b виконується за $O(n^2)$.

Розширене правило Крамера - це модифікація стандартного правила Крамера, яке дозволяє розв'язувати системи лінійних рівнянь длявироджених матриць (тобто для матриць, визначник яких дорівнює нулю).

Розраховується за формулою:

$$x_i = \frac{det(A_i)}{det(A)}$$

Код програми на мові Python:

```
import numpy as np

# Задана некоректна матриця A
A = np.array([
    [1.00, 0.99, 0.98, 0.97, 0.96],
    [0.99, 1.00, 0.99, 0.98, 0.97],
    [0.98, 0.99, 1.00, 0.99, 0.98],
    [0.97, 0.98, 0.99, 1.00, 0.99],
    [0.96, 0.97, 0.98, 0.99, 1.00]
])
```

```
# Вектор x складається з одиниць
x = np.ones(A.shape[0])

# Обчислення вектора b
b = np.dot(A, x)

# Метод Гауса-Жордана
def gauss_jordan(A, b):
    n = len(b)
    augmented_matrix = np.hstack([A, b.reshape(-1, 1)])

    for i in range(n):
        # Нормалізація рядка
        augmented_matrix[i] = augmented_matrix[i] / augmented_matrix[i, i]
        # Обнулення інших елементів в стовпці
        for j in range(n):
            if i != j:
                augmented_matrix[j] -= augmented_matrix[i] * augmented_matrix[j, i]

    return augmented_matrix[:, -1]

def lu_decomposition(A, b):
    n = len(A)
    L = np.zeros_like(A)
    U = np.zeros_like(A)

    for i in range(n):
        for k in range(i, n):
            U[i, k] = A[i, k] - sum(L[i, j] * U[j, k] for j in range(i))
```

```python
        for k in range(i, n):
            if i == k:
                L[i, i] = 1  # одиниці на діагоналі
            else:
                L[k, i] = (A[k, i] - sum(L[k, j] * U[j, i] for j in range(i))) / U[i, i]

    y = np.zeros_like(b)
    for i in range(n):
        y[i] = b[i] - sum(L[i, j] * y[j] for j in range(i))

    x = np.zeros_like(b)
    for i in range(n - 1, -1, -1):
        x[i] = (y[i] - sum(U[i, j] * x[j] for j in range(i + 1, n))) / U[i, i]

    return x

x_gauss_jordan = gauss_jordan(A.copy(), b.copy())
x_lu_decomposition = lu_decomposition(A.copy(), b.copy())

print(x_gauss_jordan, x_lu_decomposition)
```

Код правила Крамера:

```
import numpy as np

# Функція для обчислення визначників та розв'язання за правилом Крамера
def cramer_rule(A, b):
    det_A = np.linalg.det(A)  # Визначник основної матриці A
    if abs(det_A) < 1e-10:
        return "Матриця вироджена, розв'язок неможливий за правилом Крамера."

    n = len(b)
    x = np.zeros(n)

    for i in range(n):
        # Створюємо копію матриці A та замінюємо i-й стовпець на вектор b
        A_i = A.copy()
        A_i[:, i] = b
        det_A_i = np.linalg.det(A_i)  # Визначник зміненої матриці
        x[i] = det_A_i / det_A  # Знаходимо x_i

    return x

# Виконання перевірки правила Крамера
A = np.array([
    [1.00, 0.99, 0.98, 0.97, 0.96],
    [0.99, 1.00, 0.99, 0.98, 0.97],
    [0.98, 0.99, 1.00, 0.99, 0.98],
    [0.97, 0.98, 0.99, 1.00, 0.99],
    [0.96, 0.97, 0.98, 0.99, 1.00]
])
```

```
x = np.ones(A.shape[0])

b = np.dot(A, x)

x_cramer = cramer_rule(A, b)
print(x_cramer)
```

Результат роботи програм:

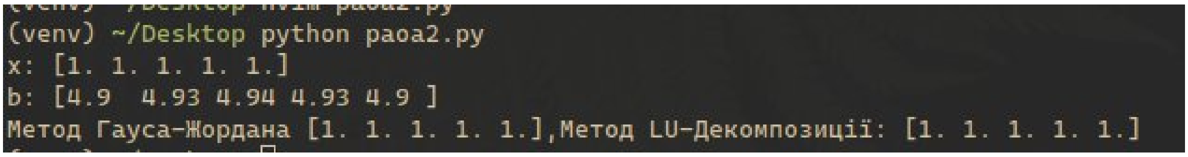

Рис 1.1 – Результат роботи метода Гауса-Жордана та декомпозиції

Рис 1.2 – Результат виконання програми правила Крамера

Висновки:

Під час виконання лабораторної роботи було розглянуто два класичні методи розв'язання систем лінійних алгебраїчних рівнянь (СЛАР): метод Гауса-Жордана та метод LU-розкладу. Обчислення показали, що для заданої матриці A і вектора b, обчисленого на основі вектора xxx, який складається з одиниць, обидва методи дали однаковий результат. Це свідчить про правильність реалізації алгоритмів та їхню здатність працювати з погано обумовленими матрицями.Було використано модифіковане розширене правило Крамера для перевірки розв'язку. Результати показали, що правило Крамера також дало правильний розв'язок. Робота дозволила закріпити знання з алгоритмів розв'язання СЛАР. Реалізовані алгоритми показали коректність та ефективність обчислень, що підтверджується збігом результатів різних підходів.

Використані джерела:

Чисельні методи розв'язання систем лінійних алгебраїчних рівнянь на ЕОМ. [Електронний ресурс].– Режим доступу: https://web.posibnyky.vntu.edu.ua/fksa/14moskvina__komp_metod_dosl_analiz_danykh/lek2.htm

Чисельні методи розв’язання систем лінійних алгебраїчних рівнянь. [Електронний ресурс]. – Режим доступу: https://dspace.onu.edu.ua/bitstream/123456789/27785/1/numerical%20methods.pdf

Прямі та ітераційні методи розв'язання систем лінійних рівнянь. [Електронний ресурс]. – Режим доступу: https://pns.hneu.edu.ua/pluginfile.php/293296/mod_resource/content/2/%D0%A2%D0%B5%D0%BC%D0%B0%203.pdf