Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут комп’ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

Звіт
Про виконання лабораторної роботи №3

На тему:

«Розв’язування систем лінійних алгебраїчних рівнянь методом Крамера та методом оберненої матриці»
з дисципліни «Чисельні методи»

Лекторка:
доцент каф. ПЗ
Мельник Н. Б.
Виконав:
ст. гр. ПЗ-11
Ясногородський Н.В.
Прийняла:
доцент каф. ПЗ
Мельник Н. Б.

« __ » _____ 2022 р.

∑ = _____ _____ .

Львів – 2022

Тема: Розв’язування систем лінійних алгебраїчних рівнянь методом Крамера та методом оберненої матриці.

Мета: Ознайомлення на практиці з методом Крамера та методом оберненої матриці розв’язування систем лінійних алгебраїчних рівнянь.

Теоретичні відомості

Метод Крамера – використовується для розв’язання СЛАР яка містить n рівнянь та n невідомих, причому визначник матриці коефіцієнтів A не дорівнює нулю. Для знаходження коренів застосовують формулу $x_i = \frac{\det \mathrm{A_i}}{\det \mathrm{A}}$, де A_i це матриця A, в якій *i*-тий стовпець замінений стовпцем вільних членів (матриці B).

Метод оберненої матриці – полягає в отриманні рівняння $X = A^{-1}B$ з рівняння $AX = B$ шляхом домноження його на A^{-1}. Для знаходження оберненої матриці потрібно транспонувати матрицю алгебраїчних доповнень та поділити її на визначник матриці A. Для знаходження алгебраїчних доповнень скористаємось формулою $\bar{A}_{ij} = (-1)^{i+j} M_{ij}$, де M_{ij} – це мінор, який отримують з матриці A викреслюванням *i*-го рядка та *j*-го стовпця.

Індивідуальне завдання

Варіант 15

Скласти програму розв’язування системи лінійних алгебраїчних рівнянь методом оберненої матриці та методом Крамера:

$$\begin{cases} 0{,}83x_1 + 1{,}41x_2 - 0{,}58x_3 = 2{,}71 \\ 1{,}23x_1 + 0{,}83x_2 + 1{,}17x_3 = 5{,}26 \\ 1{,}43x_1 - 1{,}58x_2 + 0{,}83x_3 = 1{,}03 \end{cases}$$

Код функцій

```
matrix = [
    [0.83, 1.41, 0.58],
    [1.23, 0.83, 1.17],
    [1.43, 1.58, 0.83],
]
constants = [2.71, 5.26, 1.03]

def det(m, nOrder):
    if nOrder == 1:
        return m[0][0]
    z = 0
    for r in range(nOrder):
        k = m[:]
        del k[r]
        z += m[r][0] * (-1) ** r * det([p[1:] for p in k], nOrder - 1)
    return z

def cramer(mtrx, values):
    nOrder = len(values)
    main_det = det(mtrx, nOrder)

    if main_det == 0:
        return []

    res = []

    for i in range(nOrder):
        mtrx_with_changed_column = [
            r[:i] + [s] + r[i + 1 :] for r, s in zip(mtrx, values)
        ]
        res.append(det(mtrx_with_changed_column, nOrder) / main_det)

    return res


def matrix_multply(A, B):
    rows_A = len(A)
    cols_A = len(A[0])
    rows_B = len(B)
    cols_B = len(B[0])

    if cols_A != rows_B:
        raise ValueError("One of the matrices is not eligible for
multiplication")

    res = [[0 for _ in range(cols_B)] for _ in range(rows_A)]

    for i in range(rows_A):
        for j in range(cols_B):
            for k in range(cols_A):
                res[i][j] += A[i][k] * B[k][j]
```

```
    return res


def matrix_method(eq_matrix, values):
    width = len(values)
    determinant = det(eq_matrix, width)
    # algebraic complements
    ac_matrix = []

    if determinant == 0:
        return []

    for row in range(width):
        ac_row = []

        for column in range(width):
            intermediate = eq_matrix.copy()

            # delete row and column
            intermediate.pop(row)
            intermediate = [
                list(x)
                for x in zip(
                    *[d for i, d in enumerate(zip(*intermediate)) if i != column]
                )
            ]

            minor = det(intermediate, width - 1)
            ac_row.append(pow(-1, row + column) * minor)

        ac_matrix.append(ac_row)

    ac_transposed_matrix = [list(x) for x in zip(*ac_matrix)]
    ac_inverted_matrix = [[z / determinant for z in y] for y in ac_transposed_matrix]

    dot_product = matrix_multply(ac_inverted_matrix, [[v] for v in values])

    return [v[0] for v in dot_product]


if __name__ == "__main__":
    print(
        f"""
            Cramer method:
                x values = {cramer(matrix, constants)}

            Matrix method:
                x values = {matrix_method(matrix, constants)}
        """
    )
```

Протокол роботи

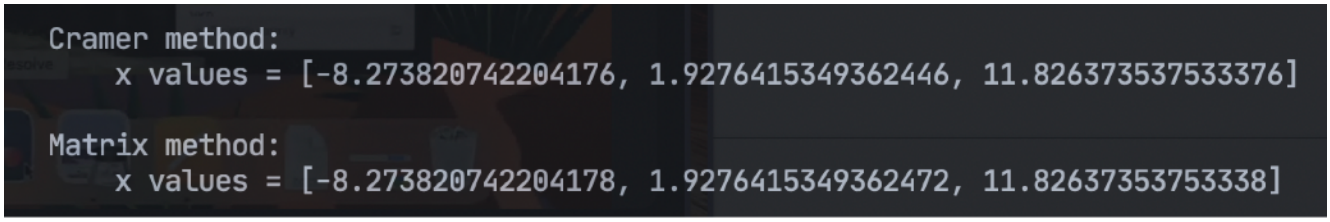

Рис.1. Робота програми

Висновки

Виконуючи лабораторну роботу №3, я навчився розв’язувати СЛАР методами Крамера та оберненої матриці, а також склав програму, яка їх розв’язує автоматично.