Міністерство освіти і науки

Національний університет „Львівська політехніка”



**Звіт**

з лабораторної роботи №3

з дисципліни: “ Чисельні методи”

Виконав:

Ст. гр. ІР-25

Баланик Б. В.

**Львів**

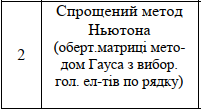
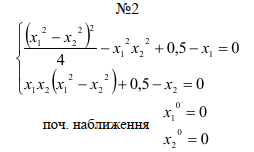
**2023**

# Мета роботи

Вивчити та застосувати метод Ньютона для знаходження коренів системи рівнянь..

# Завдання до лабораторної роботи.

**Варіант 2 Група 1**

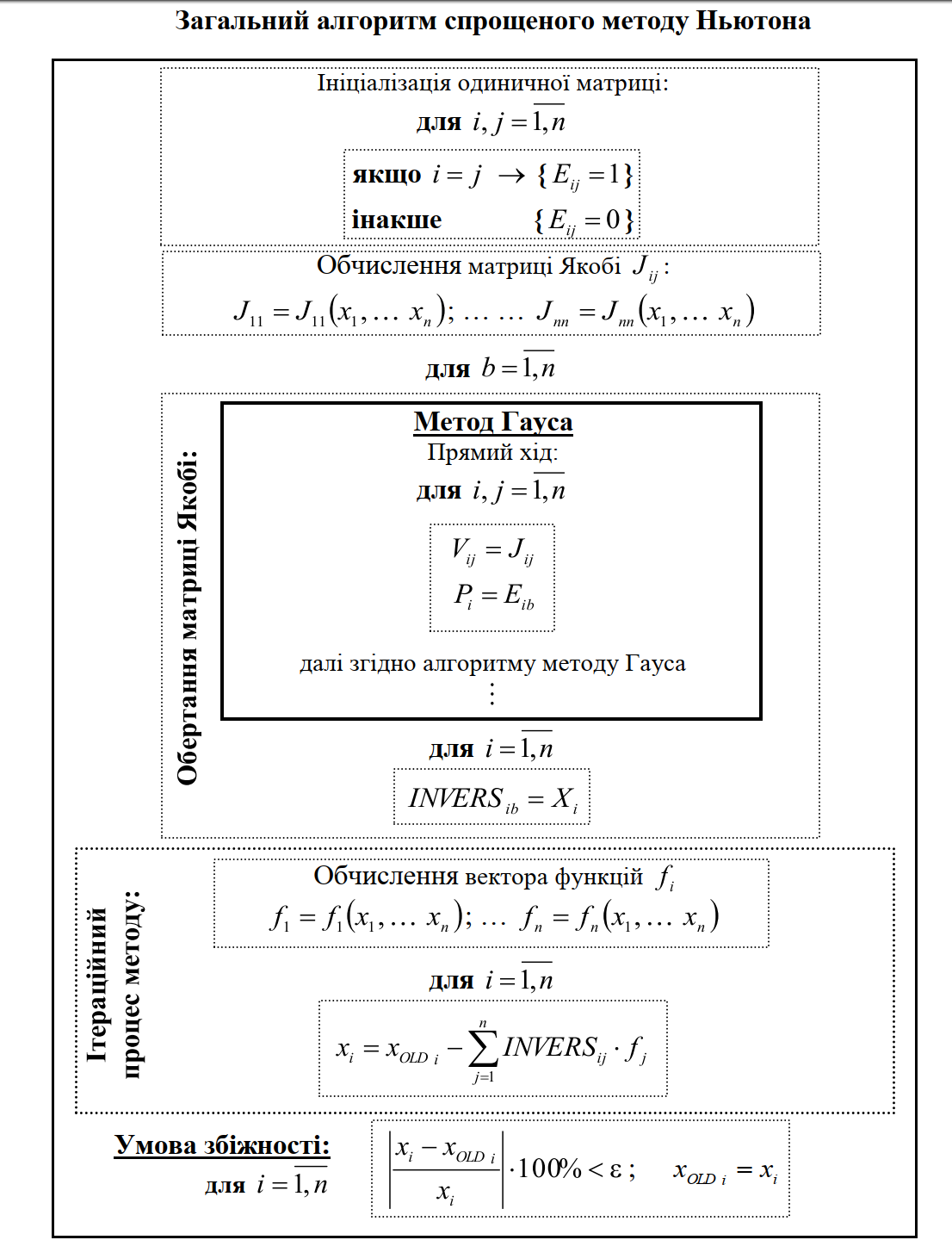
За допомогою спрощеного методу Ньютона знаходити корені системи рівнянь. 

# Короткі теоретичні відомості, що необхідні для виконання лабораторної роботи.

# Метод Ньютона (або метод дотичних) — це ітераційний чисельний метод знаходження кореня (нуля) реальної функції. Він заснований на апроксимації функції лінійною функцією та вибору перетину цієї лінії з віссю абсцис за наступне наближення.

# Блок-схема розробленої програми.

Загальний алгоритм методу поділу проміжку навпіл з пошуком ділянки локалізації:



# Список ідентифікаторів констант, змінних, функцій, методів, використаних у програмі, та їх пояснення.

* A - Матриця коефіцієнтів системи рівнянь.
* b - Вектор вільних членів.
* n - Розмірність системи рівнянь.
* x - Масив для зберігання поточних наближень розв'язків.
* newton\_method - Функція для знаходження коренів системи рівнянь за допомогою методу Ньютона.

Остаточна версія програми.

import numpy as np

def gauss\_elimination(A, b):

    """Розв'язок системи лінійних рівнянь методом Гаусса."""

    n = len(b)

    for i in range(n):

        # Вибір головного елементу по рядку

        max\_row = max(range(i, n), key=lambda r: abs(A[r][i]))

        A[i], A[max\_row] = A[max\_row], A[i]

        b[i], b[max\_row] = b[max\_row], b[i]

        # Обернення рядків A[i]

        for j in range(i + 1, n):

            factor = A[j][i] / A[i][i]

            b[j] -= factor \* b[i]

            for k in range(i, n):

                A[j][k] -= factor \* A[i][k]

    # Знаходження розв'язку

    x = np.zeros(n)

    for i in range(n - 1, -1, -1):

        x[i] = (b[i] - sum(A[i][j] \* x[j] for j in range(i + 1, n))) / A[i][i]

    return x

def system(x):

    """Визначення системи рівнянь."""

    x1, x2 = x

    f1 = (x1\*\*2 - x2\*\*2)\*\*2 / 4 - x1\*\*2 + x2\*\*2 + 0.5 - x1

    f2 = x1 \* x2 \* (x1\*\*2 - x2\*\*2) + 0.5 - x2

    return np.array([f1, f2])

def jacobian(x):

    """Обчислення матриці Якобі для системи рівнянь."""

    x1, x2 = x

    return np.array([

        [2 \* (x1\*\*2 - x2\*\*2) \* x1 / 4 - 2 \* x1 - 1, -2 \* (x1\*\*2 - x2\*\*2) \* x2 / 4 + 2 \* x2],

        [x2 \* (x1\*\*2 - x2\*\*2) + 2 \* x1 \* x2 \* x1, x1 \* (x1\*\*2 - x2\*\*2) - 2 \* x1 \* x2 \* x2 - 1]

    ])

def newton\_method(x0, eps):

    """Розв'язок системи рівнянь методом Ньютона."""

    x = x0

    while np.linalg.norm(system(x)) > eps:

        J = jacobian(x)

        F = system(x)

        delta = gauss\_elimination(J, -F)

        x += delta

    return x

def is\_solution\_valid(x):

    """Перевірка коректності розв'язку."""

    return np.linalg.norm(system(x)) < 1e-5

def main():

    # Початкова точка наближення

    x0 = np.array([0.0, 0.0])

    # Відносна похибка

    epsilon = 1e-5

    # Знаходження розв'язку системи

    solution = newton\_method(x0, epsilon)

    print("Solution:", solution)

    # Перевірка вірності розв'язку

    if is\_solution\_valid(solution):

        print("Розв'язок є правильним!")

    else:

        print("Розв'язок невірний.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Результати виконання програми.

**Solution: [0.5 0.5]**

**Розв'язок є правильним!**

Висновки.

За допомогою спрощеного методу Ньютона вдалося успішно знайти корені системи рівнянь. Отриманий результат підтверджує правильність реалізації методу та його ефективність для даної системи рівнянь.