МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ЗВІТ

про виконання лабораторної роботи №1

з теми ПРЯМІ ТА ІТЕРАЦІЙНІ МЕТОДИ РОЗВ’ЯЗУВАННЯ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ

з навчальної дисципліни: “Чисельні методи”

Виконав:

студент групи ІР-24

Шийка Андрій

Прийняв:

Сиротюк С. В.

Львів - 2025

Мета роботи:

вивчити найпоширеніші прямі та ітераційні методи розв’язування систем лінійних алгебраїчних рівнянь та способи їх застосування для обчислення визначників і обертання матриць.

Завдання:

знайти обернену матрицю методом Гауса з вибором головного елемента по всій матриці.

Короткі теоретичні відомості:

Для знаходження оберненої матриці A^(-1) для неособливої матриці A використовується співвідношення A * A^(-1) = E, де E — одинична матриця. Задача зводиться до розв'язування n систем лінійних рівнянь вигляду A * Xi = Ei, де Ei — i-й стовпець одиничної матриці, а Xi — відповідний стовпець оберненої матриці.

Метод Гауса з вибором головного елемента по всій матриці використовується для підвищення точності обчислень. На кожному кроці k серед усіх елементів підматриці (необробленої частини) обирається максимальний за модулем елемент a_mq.

Алгоритм передбачає:

1. Пошук максимального елемента у всій підматриці.

2. Перестановку рядка з головним елементом на позицію k.

3. Перестановку стовпця з головним елементом на позицію k (зі збереженням інформації про перестановку невідомих у векторі inx).

4. Прямий хід: приведення матриці до трикутного вигляду.

5. Обернений хід: знаходження невідомих.

6. Впорядкування компонент вектора розв'язку відповідно до початкового порядку невідомих.

Список ідентифікаторів констант, змінних, функцій:

- k, s: змінні для задання варіанту завдання (k - ціле, s - дійсне).

- matrix: двовимірний список, що зберігає вхідну матрицю A.

- gauss_full_pivot_solve: функція, що реалізує метод Гауса з повним вибором головного елемента.

- inverse_matrix_gauss_full: функція для знаходження оберненої матриці.
- A_in, B_in: вхідні параметри функції (матриця та вектор вільних членів).
- V: робоча копія матриці A.
- P: робоча копія вектора вільних членів (або стовпця E).
- inx: список для відстеження перестановок стовпців (невідомих).
- C: матриця коефіцієнтів після прямого ходу.
- Y: вектор проміжних результатів прямого ходу.
- X: вектор розв'язку системи.
- INV: результуюча обернена матриця.

Вхідні дані:

Матриця №1

8,3	$2,62+s$	4,1	1,9
3,92	8,45	$7,78-s$	2,46
3,77	$7,21+s$	8,04	2,28
2,21	$3,65-s$	1,69	6,69

де $s=0,02k$; $B=0,02p$; $k=$порядковому № завдання; $p=$№ групи (наприклад, для КС-21 $p=21$)

Код програми:

```
import copy


k = 12      # Номер завдання 12
s = 0.02*k  # Коефіцієнт для варіанту матриці

# Вхідні дані: Матриця №1
matrix = [
    [8.30, 2.62+s, 4.10, 1.90],
    [3.92, 8.45, 7.78-s, 2.46],
    [3.77, 7.21+s, 8.04, 2.28],
```

```
    [2.21, 3.65-s, 1.69, 6.69]
]


def gauss_full_pivot_solve(A_in, B_in):
    """
    Реалізація методу Гауса з вибором головного елемента по всій матриці.
    Алгоритм згідно з Методичкою №1, стор. 13-14 .
    """
    n = len(A_in)

    # 1. Ініціалізація вектора перестановок inx
    inx = list(range(n))

    # 2. Прямий хід. Копіювання матриць A->V, B->P
    V = copy.deepcopy(A_in)
    P = list(B_in)

    # Матриця C для коефіцієнтів (верхня трикутна)
    C = [[0.0] * n for _ in range(n)]
    # Вектор Y для зберігання результатів прямого ходу
    Y = [0.0] * n

    # Цикл по k
    for k in range(n):
        # --- Матричне сортування (пошук головного елемента) ---
        max_val = 0.0
        h = k
        w = k

        for l in range(k, n):
            for f in range(k, n):
                if abs(V[l][f]) > max_val:
                    max_val = abs(V[l][f])
                    h = l
```

```
            w = f

# Перестановка рядків (h <-> k) у V та P
# value = P_k; P_k = P_h; P_h = value
P[k], P[h] = P[h], P[k]

# value = V_kd; V_kd = V_hd; V_hd = value (для d = 1..n)
for d in range(n):
    V[k][d], V[h][d] = V[h][d], V[k][d]

# Перестановка стовпців (w <-> k) у inx, C та V
# z = inx_k; inx_k = inx_w; inx_w = z
inx[k], inx[w] = inx[w], inx[k]

for d in range(n):
    if d < k:
        # якщо d < k -> міняємо у C
        C[d][k], C[d][w] = C[d][w], C[d][k]
    else:
        # інакше -> міняємо у V
        V[d][k], V[d][w] = V[d][w], V[d][k]

# --- Обчислення (Елімінація) ---
# Y_k = P_k / V_kk
Y[k] = P[k] / V[k][k]

# для i = k+1..n: P_i = P_i - V_ik * Y_k
for i in range(k + 1, n):
    P[i] = P[i] - V[i][k] * Y[k]

# для j = k+1..n
for j in range(k + 1, n):
    # C_kj = V_kj / V_kk
    C[k][j] = V[k][j] / V[k][k]
    # V_ij = V_ij - V_ik * C_kj
```

```
            for i in range(k + 1, n):
                V[i][j] = V[i][j] - V[i][k] * C[k][j]


    # --- Обернений хід ---
    X = [0.0] * n
    # X_n = Y_n (у псевдокоді це в циклі, але база рекурсії - останній елемент)


    # Цикл від n-1 до 0
    for i in range(n - 1, -1, -1):
        sum_cx = 0.0
        for j in range(i + 1, n):
            sum_cx += C[i][j] * X[j]
        X[i] = Y[i] - sum_cx


    # --- Впорядкування X_i ---
    # Відновлення порядку коренів згідно з вектором inx
    for i in range(n):
        if inx[i] != i:
            z = inx[i]
            val = X[i]
            X[i] = X[z]
            X[z] = val

            # Оновлення вектора inx (swap logic з методички)
            inx[i] = inx[z]
            inx[z] = z


    return X


def inverse_matrix_gauss_full(A):
    """
    Знаходження оберненої матриці методом Гауса.
    Алгоритм згідно з Методичкою №1, п. 5
    з використанням схеми повного вибору головного елемента
```

```
    """
    n = len(A)
    # Матриця для результату INVERS
    INV = [[0.0] * n for _ in range(n)]

    # Для кожного стовпця b одиничної матриці E
    for b in range(n):
        # Формуємо стовпець одиничної матриці E_b
        E_vec = [0.0] * n
        E_vec[b] = 1.0

        # Розв'язуємо систему AX = E_b методом Гауса
        # (в даному випадку - з повним вибором головного елемента)
        X_col = gauss_full_pivot_solve(A, E_vec)

        # Записуємо результат у відповідний стовпець оберненої матриці
        for i in range(n):
            INV[i][b] = X_col[i]

    return INV

# Виконання програми
print(f"Вхідна матриця (k={k}, s={s:.2f}):")
for row in matrix:
    print([round(x, 4) for x in row])

inverse_m = inverse_matrix_gauss_full(matrix)

print("\nОбернена матриця:")
for row in inverse_m:
    print(" ".join(f"{x: .6f}" for x in row))

# Перевірка: A * A_inv повинна дорівнювати одиничній матриці E
print("\nПеревірка (A * A_inv):")
check = [[0.0]*4 for _ in range(4)]
```

```
for i in range(4):
    for j in range(4):
        val = sum(matrix[i][k] * inverse_m[k][j] for k in range(4))
        check[i][j] = val
for row in check:
    print(" ".join(f"{x: .6f}" for x in row))
```

Результат:

```
na_p1_labs_iot_nulp on  lab/1 [!] via 🐍 v3.12.2 (.venv)
❯ python lab1/lab1.py

Вхідна матриця (k=12, s=0.24):
[8.3, 2.86, 4.1, 1.9]
[3.92, 8.45, 7.54, 2.46]
[3.77, 7.45, 8.04, 2.28]
[2.21, 3.41, 1.69, 6.69]

Обернена матриця:
 0.154499  0.130810 -0.196182 -0.025119
-0.035131  0.701375 -0.633081 -0.032169
-0.032851 -0.643723  0.746588 -0.008407
-0.024833 -0.238100  0.198899  0.176296

Перевірка (A * A_inv):
 1.000000 -0.000000 -0.000000  0.000000
 0.000000  1.000000  0.000000  0.000000
-0.000000 -0.000000  1.000000  0.000000
-0.000000  0.000000 -0.000000  1.000000
```

Висновок:

У ході виконання лабораторної роботи було вивчено та програмно реалізовано прямий метод розв’язування систем лінійних алгебраїчних рівнянь — метод Гауса з вибором головного елемента по всій матриці. Цей підхід було застосовано для виконання завдання №12: знаходження оберненої матриці для заданого варіанту.

Аналіз отриманих результатів показав, що реалізований алгоритм успішно виконав LU-подібне перетворення вхідної матриці з використанням матричного сортування. Скалярний добуток вихідної матриці A на знайдену обернену матрицю A^(-1) дав у результаті одиничну матрицю E (з урахуванням машинної точності), що підтверджує правильність роботи програми та високу точність методу повного вибору головного елемента.