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

Звіт
Про виконання лабораторної роботи №4

На тему:

«Розв’язування систем лінійних алгебраїчних рівнянь методом Гауса та методом LU-розкладу»
з дисципліни «Чисельні методи»

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

« __ » _____ 2022 р.

∑ = _____ _____ .

Львів – 2022

Тема: Розв’язування систем лінійних алгебраїчних рівнянь методом Гауса та методом LU-розкладу.

Мета: Ознайомлення на практиці з методом Гауса та методом LU-розкладу розв’язування систем лінійних алгебраїчних рівнянь.

Теоретичні відомості

Метод Гауса – поділяється на два етапи – прямий і зворотній хід. Під час прямого ходу СЛАР зводиться до східчастого вигляду. Для того щоб не втрачати точність при діленні на діагональний елемент, існує модифікація цього методу – з вибором головного елемента. Це означає, що на кожному етапі зведення системи до східчастого вигляду рівняння переставляють таким чином, щоб на діагоналі знаходився найбільший за модулем елемент цього стовпця. Під час зворотного ходу застосовують формулу $x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{k=i+1}^{n} a_{ik}x_k \right), \quad i = \overline{1,n}$ для знаходження коренів.

Метод LU-розкладу – полягає в розкладанні матриці коефіцієнтів A на добуток нижньої трикутної матриці L, елементи головної діагоналі якої не дорівнюють нулеві та верхньої трикутної U, на головній діагоналі якої містяться одиниці. Для знаходження оберненої матриці потрібно транспонувати матрицю алгебраїчних доповнень та поділити її на визначник матриці A. Звідси отримаємо рівняння LUX=B, введемо допоміжний вектор Y=UX і визначимо його з рівняння LY=B за формулою $y_i = \frac{1}{l_{ii}}\left(b_i - \sum_{m=1}^{i-1} l_{im}y_m \right), \quad i = \overline{1,n}$, що буде прямим ходом цього методу. Під час зворотного ходу визначаємо корені за формулою

$$x_i = y_i - \sum_{m=i+1}^{n} u_{im}x_m, \quad i \le n.$$

Для знаходження матриць L і U скористаємось формулами (метод Краута):

$$l_{ik} = a_{ik} - \sum_{m=1}^{k-1} l_{im}u_{mk}, \quad u_{ii} = 1,\ u_{kj} = \frac{1}{l_{kk}}\left(a_{kj} - \sum_{m=1}^{k-1} l_{km}u_{mj} \right),$$

$$i = \overline{k,n}, \quad j = \overline{k+1,n}$$

Індивідуальне завдання

Варіант 15

Скласти програму розв’язування системи лінійних алгебраїчних рівнянь методом Гауса та LU-розкладу:

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

Код функцій

```
from math import fabs

import numpy as np

A = np.array(
    [
        [0.83, 1.41, 0.58],
        [1.23, 0.83, 1.17],
        [1.43, 1.58, 0.83],
    ]
)

B = np.array([2.71, 5.26, 1.03])


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

    # Elimination phase
    for k in range(n - 1):
        # find the best raw
        for i in range(k + 1, n):
            if fabs(A[i, k]) > fabs(A[k, k]):
                A[[k, i]] = A[[i, k]]
                B[[k, i]] = B[[i, k]]
                break

        for i in range(k + 1, n):
            if A[i, k] != 0.0:
```

```
                cof = A[i, k] / A[k, k]
                # calculate the new row
                A[i] = A[i] - cof * A[k]
                # update vector b
                B[i] = B[i] - cof * B[k]
        print(f"\nIteration {k}:")
        _print_a_and_b(A, B)

    # backward substitution
    x = np.zeros(n)
    for k in range(n - 1, -1, -1):
        x[k] = (B[k] - np.dot(A[k, k + 1 : n], x[k + 1 : n])) / A[k, k]

    return x


def lu_solve(L, U, P, B):
    def forward_sub(L, b):
        """solution to Lx = b
        L must be a lower-triangular matrix
        b must be a vector of the same leading dimension as L
        """
        n = len(L)
        x = np.zeros(n)
        for i in range(n):
            sum = 0
            for j in range(i):
                sum += L[i, j] * x[j]
            x[i] = (b[i] - sum) / L[i, i]
        return x

    def back_sub(U, b):
        """solution to Ux = b
        U must be an upper-triangular matrix
        b must be a vector of the same leading dimension as U
        """
        n = len(U)
        x = np.zeros(n)
        for i in range(n - 1, -1, -1):
            sum = 0
            for j in range(n - 1, i, -1):
                sum += U[i, j] * x[j]
            x[i] = (b[i] - sum) / U[i, i]
        return x
```

```
    y = forward_sub(L, np.dot(P, B))
    x = back_sub(U, y)
    return x


def lu_decompose(A):
    """
    Decomposes a matrix A by PA=LU
    """

    def pivotize(m):
        """
        Creates the pivoting matrix for m.
        """
        n = len(m)
        ID = [[float(i == j) for i in range(n)] for j in range(n)]
        for j in range(n):
            row = max(range(j, n), key=lambda i: abs(m[i][j]))
            if j != row:
                ID[j], ID[row] = ID[row], ID[j]
        return ID

    n = len(A)
    L = np.zeros((n, n))
    U = np.zeros((n, n))
    P = np.asarray(pivotize(A))
    PA = np.dot(P, A)
    for j in range(n):
        L[j][j] = 1.0
        for i in range(j + 1):
            s1 = sum(U[k][j] * L[i][k] for k in range(i))
            U[i][j] = PA[i][j] - s1
        for i in range(j, n):
            s2 = sum(U[k][j] * L[i][k] for k in range(j))
            L[i][j] = (PA[i][j] - s2) / U[j][j]

    return L, U, P


def _print_a_and_b(a, b):
    print("A:")
    print_matrix(a)
    print("B:")
    print(b)
```

```
def print_matrix(mtrx):
    s = [[str(e) for e in row] for row in mtrx]
    lens = [max(map(len, col)) for col in zip(*s)]
    fmt = "\t".join("{{:{}}}".format(x) for x in lens)
    table = [fmt.format(*row) for row in s]
    print("\n".join(table))


if __name__ == "__main__":
    print("Initial values:")
    _print_a_and_b(A, B)

    print("\n\nGause method:")
    x = gauss_elim(A.copy(), B.copy())
    print("\nResulting vector x:")
    print(x)
    print("\nVerifying results: AX - B = ", np.dot(A, x) - B)

    print("\n\nLU method:")
    L, U, P = lu_decompose(A.copy())

    print("\nL:")
    print_matrix(L)
    print("\nU:")
    print_matrix(U)
    print("\nP:")
    print_matrix(P)
    print("\nIs LU = PA:", np.array_equal(np.dot(L, U), np.dot(P, A)))

    x = lu_solve(L, U, P, B.copy())
    print("\nResulting vector x:")
    print(x)

    print("\nVerifying results: AX - B = ", np.dot(A, x) - B)
```

Протокол роботи

```
Initial values:
A:
0.83    1.41    0.58
1.23    0.83    1.17
1.43    1.58    0.83
B:
[2.71 5.26 1.03]


Gause method:

Iteration 0:
A:
1.23    0.83                    1.17
0.0     0.8499186991869918      -0.20951219512195118
0.0     0.6150406504065041      -0.5302439024390243
B:
[ 5.26       -0.83943089 -5.08528455]

Iteration 1:
A:
1.23    0.83                    1.17
0.0     0.8499186991869918      -0.20951219512195118
0.0     0.0                     -0.3786311459728333
B:
[ 5.26       -0.83943089 -4.47783337]

Resulting vector x:
[-8.27382074  1.92764153 11.82637354]

Verifying results: AX - B =  [ 0.00000000e+00 -8.88178420e-16  6.66133815e-16]


LU method:

L:
1.0                     0.0                     0.0
0.5804195804195804      1.0                     0.0
0.8601398601398602      -1.0732018726060442     1.0

U:
1.43    1.58                    0.83
0.0     0.49293706293706285     0.09825174825174826
0.0     0.0                     0.5615278762945098

P:
0.0     0.0     1.0
1.0     0.0     0.0
0.0     1.0     0.0

Is LU = PA: True

Resulting vector x:
[-8.27382074  1.92764153 11.82637354]

Verifying results: AX - B =  [0.00000000e+00 8.88178420e-16 2.22044605e-16]
```

Рис.1. Робота програми

Висновки

Виконуючи лабораторну роботу №4, я навчився розв’язувати СЛАР методами Гауса та LU-розкладу, а також склав програму, яка їх розв’язує автоматично.