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

Солтисюк Д. А.

**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 для знаходження коренів.

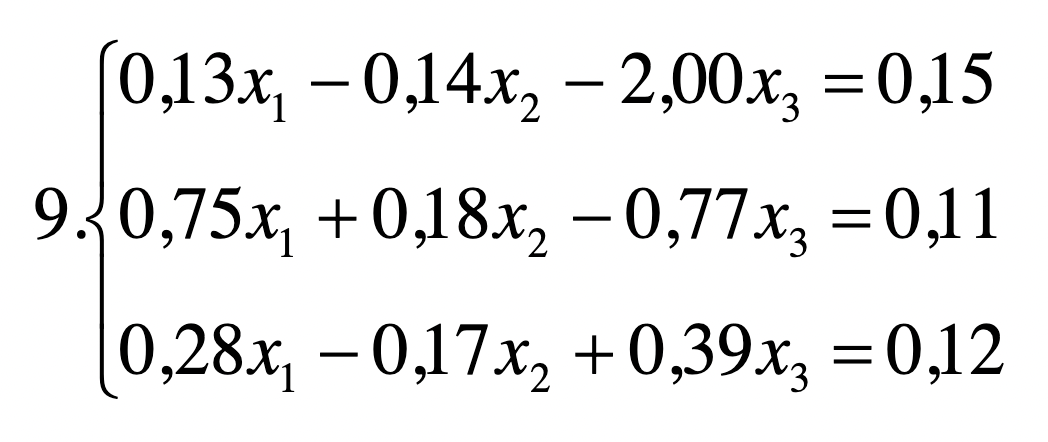
**Метод LU-розкладу** – полягає в розкладанні матриці коефіцієнтів А на добуток нижньої трикутної матриці L, елементи головної діагоналі якої не дорівнюють нулеві та верхньої трикутної U, на головній діагоналі якої містяться одиниці. Для знаходження оберненої матриці потрібно транспонувати матрицю алгебраїчних доповнень та поділити її на визначник матриці А. Звідси отримаємо рівняння LUX=B, введемо допоміжний вектор Y=UX і визначимо його з рівняння LY=B за формулою , що буде прямим ходом цього методу. Під час зворотного ходу визначаємо корені за формулою

.

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

**Індивідуальне завдання**

*Варіант 15*

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

**Код функцій**

**from** math **import** fabs

**import** numpy **as** np

**from** common.utils **import** print\_header, print\_matrix, print\_method\_introduction

Row = list[float]

Matrix = list[Row]

**class** Method:

**def** compile(self):

**raise** NotImplementedError

**def** execute\_method(self):

**raise** NotImplementedError

**class** MatrixOrientedMethod(Method):

A = np.array([])

B = np.array([])

method\_name: str = "Unknown"

**def** \_\_init\_\_(self, A: Matrix, B: Row) -> **None**:

super().\_\_init\_\_()

self.A = np.array(A)

self.B = np.array(B)

**def** execute\_method(self):

**raise** NotImplementedError

**def** compile(self):

print\_method\_introduction(self.method\_name)

print\_header("Initial values")

print\_matrix(self.A, "A")

print\_matrix([self.B], "B")

result = self.execute\_method()

**print**("\n")

print\_header("Resulting vectors")

**print**(result)

**print**("\n")

print\_header("Results verification")

**print**(f"AX - B = {np.dot(self.A, result) - self.B}")

**class** GaussianEliminationMethod(MatrixOrientedMethod):

**def** \_\_init\_\_(self, A: Matrix, B: Row) -> **None**:

super().\_\_init\_\_(A, B)

self.method\_name = "Gaussian elimination"

**def** execute\_method(self):

n = len(self.B)

*# Elimination phase*

**for** k **in** range(n - 1):

*# find the best raw*

**for** i **in** range(k + 1, n):

**if** fabs(self.A[i, k]) > fabs(self.A[k, k]):

self.A[[k, i]] = self.A[[i, k]]

self.B[[k, i]] = self.B[[i, k]]

**break**

**for** i **in** range(k + 1, n):

**if** self.A[i, k] != 0.0:

cof = self.A[i, k] / self.A[k, k]

*# calculate the new row*

self.A[i] = self.A[i] - cof \* self.A[k]

*# update vector b*

self.B[i] = self.B[i] - cof \* self.B[k]

**print**(f"\nIteration {k}:")

print\_matrix(self.A, "A")

print\_matrix([self.B], "B")

*# backward substitution*

x = np.zeros(n)

**for** k **in** range(n - 1, -1, -1):

x[k] = (self.B[k] -

np.dot(self.A[k, k + 1:n], x[k + 1:n])) / self.A[k, k]

**return** x

**class** LUDecompositionMethod(MatrixOrientedMethod):

**def** \_\_init\_\_(self, A: Matrix, B: Row) -> **None**:

super().\_\_init\_\_(A, B)

self.method\_name = "LU decomposition"

**def** lu\_decompose(self):

"""

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(self.A)

L = np.zeros((n, n))

U = np.zeros((n, n))

P = np.asarray(pivotize(self.A))

PA = np.dot(P, self.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** lu\_solve(self, 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** execute\_method(self):

L, U, P = self.lu\_decompose()

print\_matrix(L, "L")

print\_matrix(U, "U")

print\_matrix(P, "P")

**print**("\nIs LU = PA:", np.array\_equal(np.dot(L, U), np.dot(P, self.A)))

**return** self.lu\_solve(L, U, P, self.B.copy())

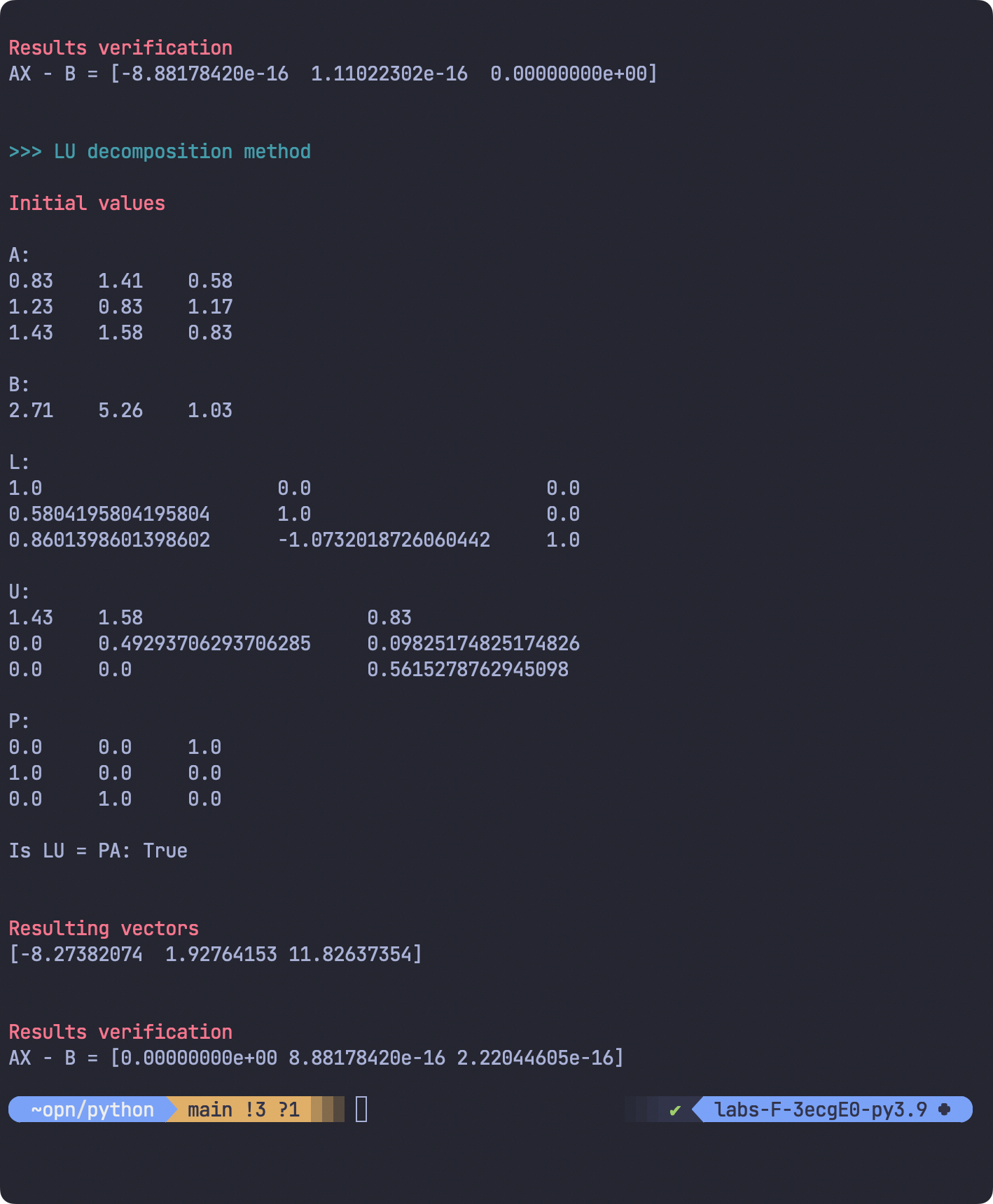
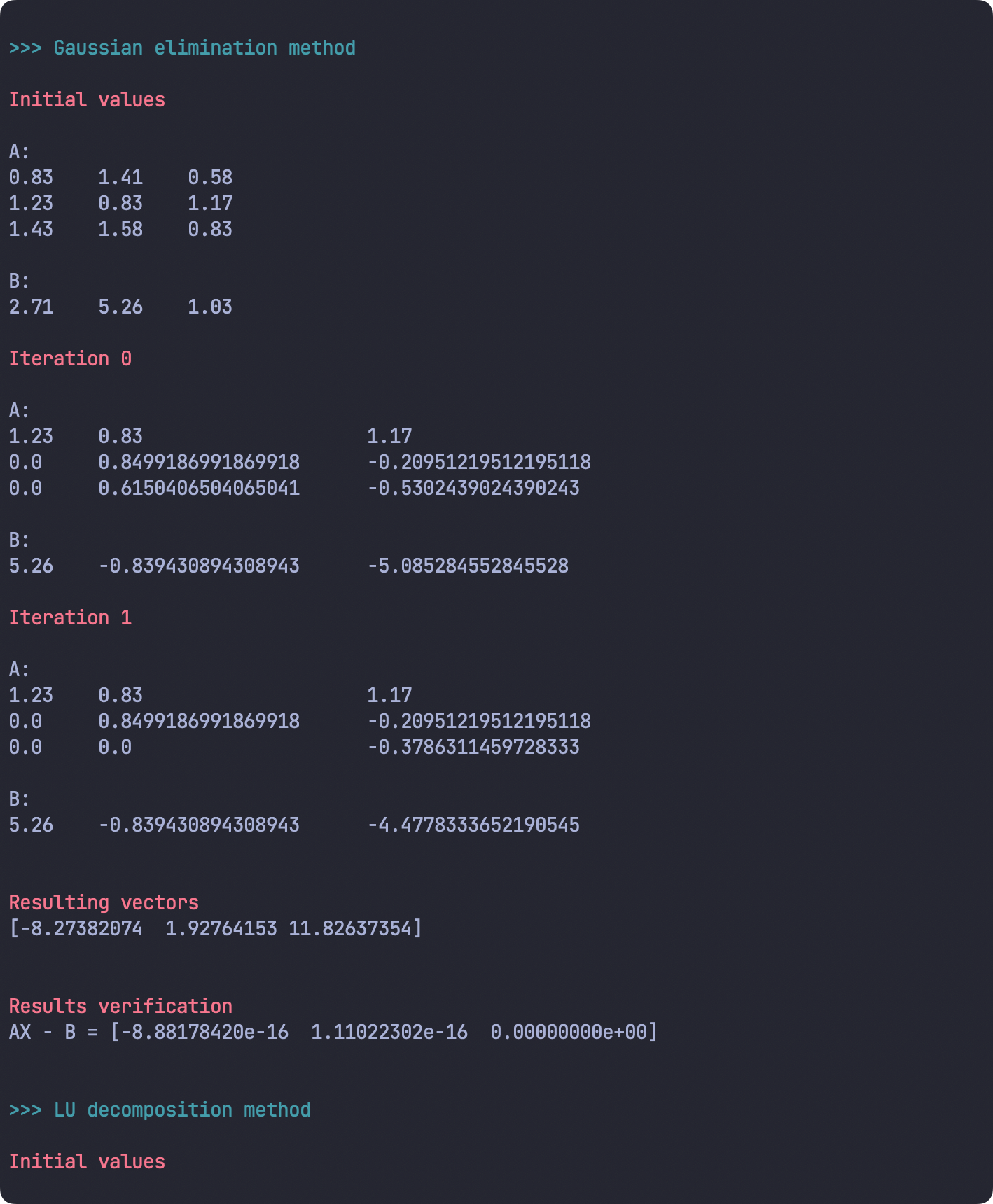
**Протокол роботи**

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.