НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи
з дисципліни «АЛГОРИТМИ І СИСТЕМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ 1.МАТЕМАТИЧНІ АЛГОРИТМИ»
на тему
"Розв'язання систем лінійних рівнянь"

Виконав:
студент групи КМ-01
Романецький М.С.

Перевірила:
Асистент кафедри ПМА
Ковальчук-Химюк Л. О.

Київ — 2023

Зміст

Вступ

Метою роботи є вивчити правила використання програмних засобів для факторизації матриць і розв’язання систем лінійних алгебраїчних рівнянь, розв’язувати задану систему рівнянь, провести порівняльний аналіз вивчених чисельних методів розв’язання СЛР.

Основна частина

Варіант 1

Варіант	Матриця коефіцієнтів системи A				Вектор вільних членів B
1	4	3	2	1	3
	3	6	4	2	6
	2	4	6	3	4
	1	2	3	4	7

Вимоги до ПЗ

1. Реалізувати перевірки на некоректний ввід (буква замість числа, порожнє введення, символи замість числа, занадто великі числа).
2. У графічному інтерфейсі повинно бути передбачено можливість гнучкого налагодження розмірності розв'язуваної задачі.
3. Має зберігатись логічно правильне розв'язання (нижня межа інтегрування не має перевищувати верхню, тощо)

Алгоритми розв'язання

Програма складається з графічного інтерфейсу, та методів розв'язання на мові Python та мові Octave.

- Мовою Python реалізовано розв'язання системи рівнянь LU методом.
- Мовою Octave реалізовано розв'язання системи рівнянь LU методом.

Алгоритм LU працює наступним чином:

1. Розкладання LU: Матриця A розкладається на дві матриці: L (нижньотрикутна матриця) та U (верхньотрикутна матриця). Тобто, якщо A - це наша матриця, то A = L * U.
2. Передувальний прохід: Множимо матрицю L на вектор b, ми отримуємо новий вектор. Цей крок зводить нашу систему до верхньотрикутної форми.
3. Зворотній прохід: Використовуючи матрицю U та новий вектор, ми отримуємо розв'язок системи.

Цей метод ефективний для розв'язання великих систем лінійних рівнянь

Висновки

На мові Python:

- Час виконання — 0.002 сек
- Похибка — 0
- Результат — [0. 1. 5. -10.]

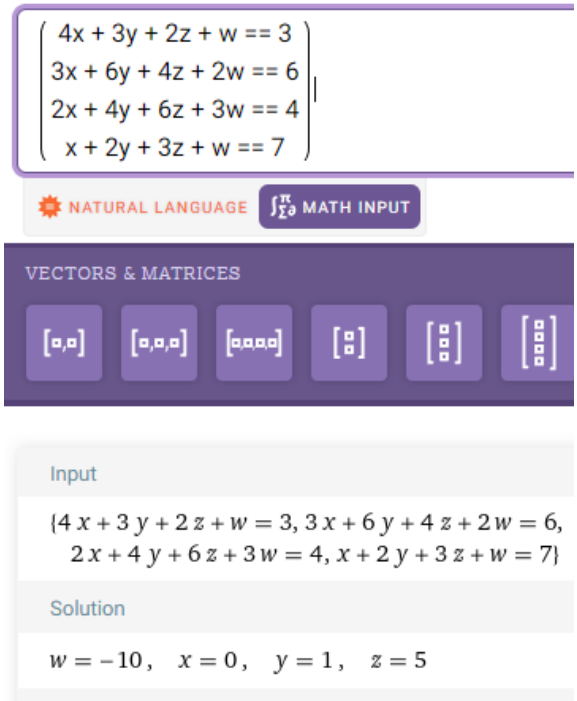
На мові Octave:

- Час виконання — 3.10 сек
- Похибка — 0
- Результат — [0. 1. 5. -10.]

Результати співпали. Час виконання менший при використання Python. Нульова похибка скоріше за все пов'язана з доцільним використанням LU- методу для заданої матриці.

Розв’язавши систему у Wolfram Alpha бачимо, що було отримано точний розв’язок

Скріншоти виконання програми наведені в Додатку А.



Відповіді на контрольні питання

Відповіді наведені в репозиторії GitHub за [посиланням]

Використана література

1. Python to GNU Octave Bridge — https://oct2py.readthedocs.io/en/latest/
2. Python GUI Programming With Tkinter — https://realpython.com/python-gui-tkinter/

Додаток А – скріншоти роботи програми

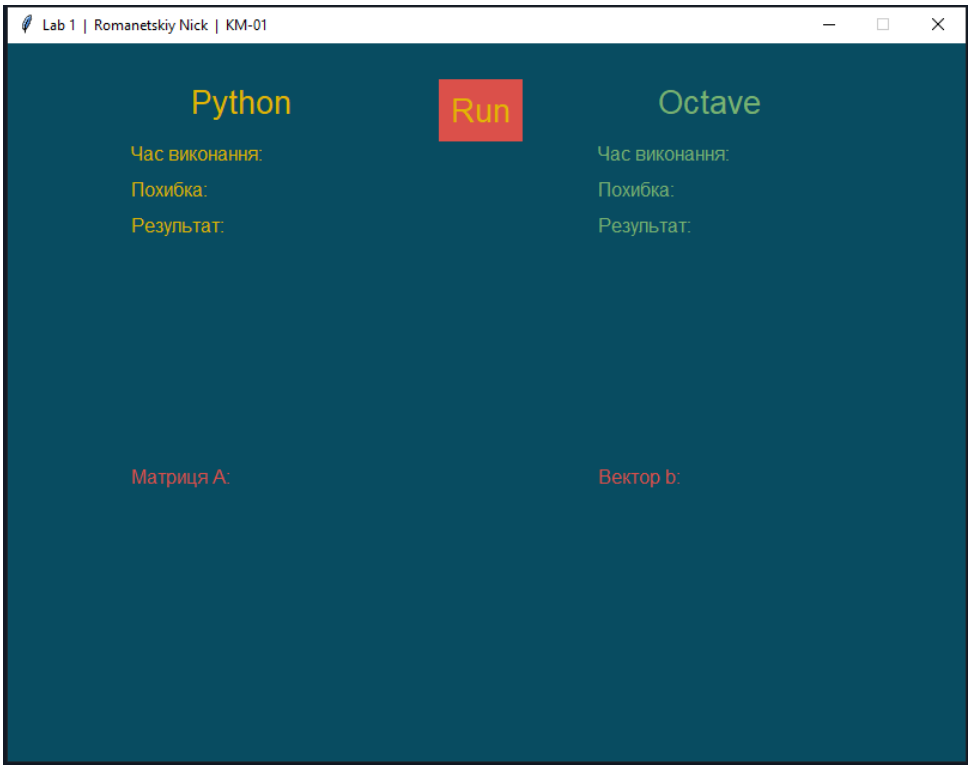


Рис. 1 – Вікно програми (до натискання кнопки ‘Run’)

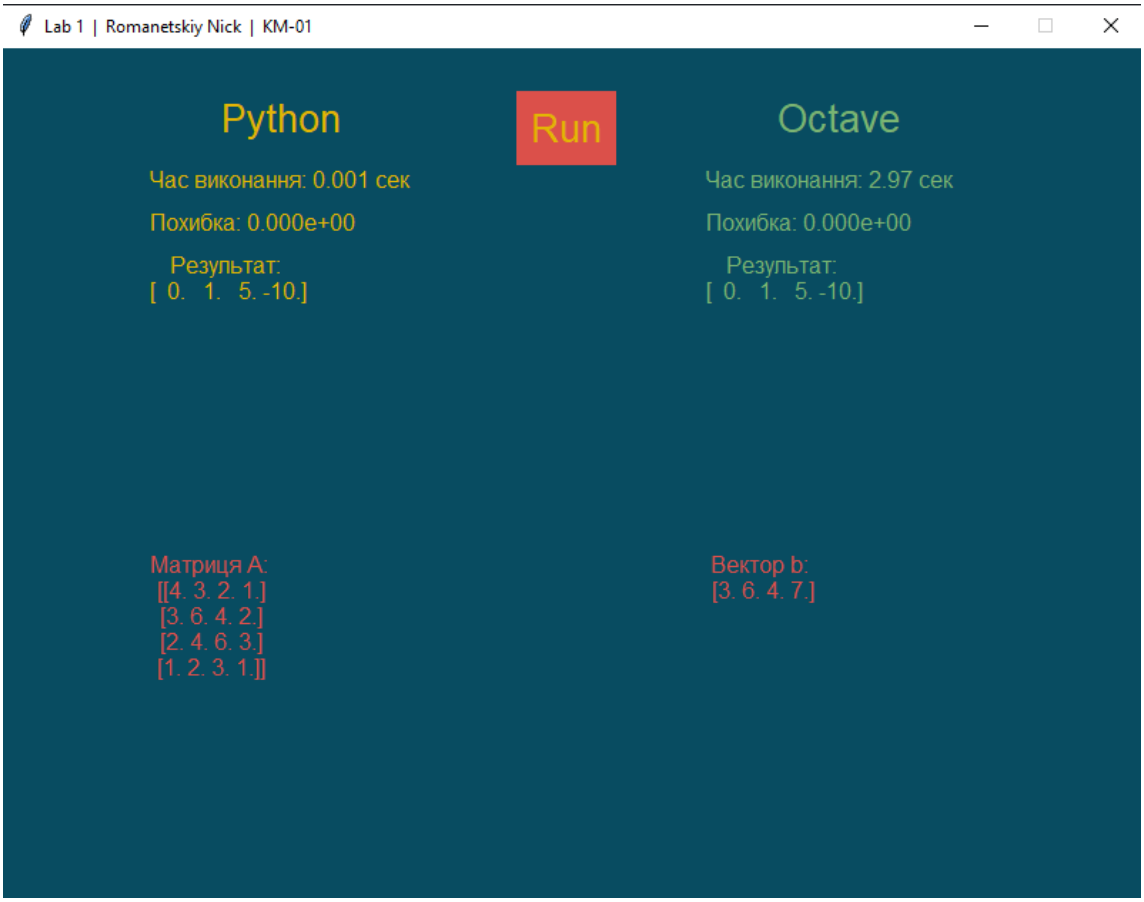


Рис. 2 – Вікно програми (після натискання кнопки ‘Run’)

Додаток Б – Блок-схема роботи програми

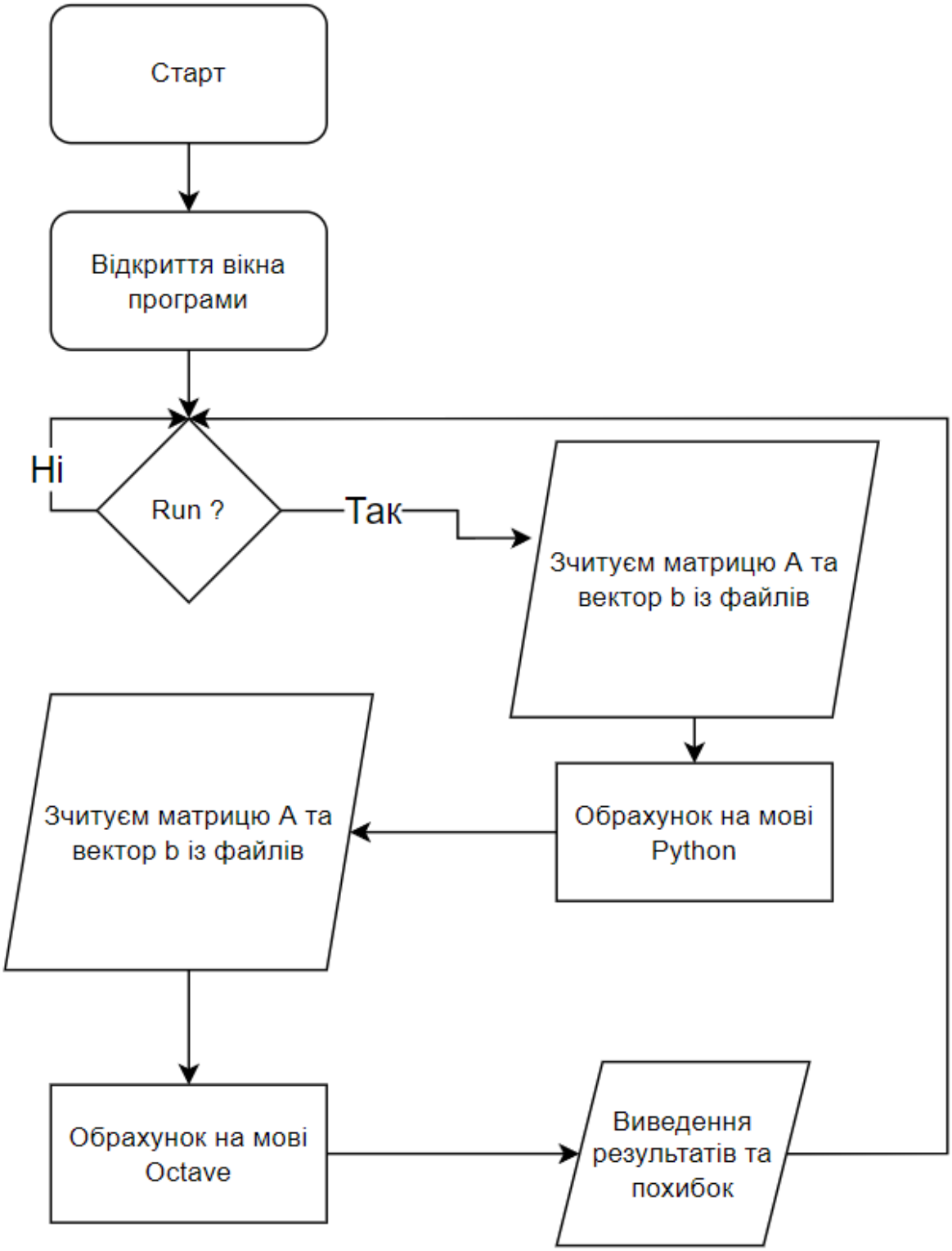


Рис. 3 – Блок-схема роботи програми

Додаток В – Код програми

Вміст файлу Matrix_A.txt :

```
4 3 2 1
3 6 4 2
2 4 6 3
1 2 3 1
```

Вміст файлу Vector_b.txt :

```
3
6
4
7
```

Вміст файлу main.py :

```
from window import *
import os


if __name__ == '__main__':
    os.system('cls')  #  Очищення консолі
    root.mainloop()  #  Запуск головного циклу програми

    #            ТЕСТИ

    # 1. Матриця A - буква           |  +
    # 2. Матриця A - замало чисел    |  +
    # 3. Матриця A - забагато чисел  |  +
    # 4. Матриця A - не квадратна    |  +
    # 5. Вектор b  - буква           |  +
    # 6. Вектор b  - замало чисел    |  +
    # 7. Вектор b  - забагато чисел  |  +
    # 8. Вектор b  - більше 1 числа  |  +
```

Вміст файлу window.py :

```
from octave_ import run_octave
from python_ import run_python


import time
import os
import tkinter as tk


CL_BLUE = '#084C61'
```

```
CL_RED = '#DB504A'
CL_YELLOW = '#E3B505'
CL_GRAY = '#4F6D7A'
CL_OCEAN = '#56A3A6'
CL_GREEN = '#79B473'
CL_PURPLE = '#414073'


root = tk.Tk()  #  Створення головного вікна програми
root.title('Lab 1  |  Romanetskiy Nick  |  KM-01')  #  Задання назви вікну
root.geometry('800x600')  #  Задаємо розмір вікна
root.resizable(width=False, height=False)  #  Заборона зміни розмірів вікна
root.configure(bg=CL_BLUE)  #  Задаємо колір фону


def run():
    os.system('cls')  #  Очищення консолі
    # - - - - - #
    try:
        start_python_time = time.time()
        matrix_A, vector_b, python_answer, python_accuracy = run_python()
        python_run_time = time.time() - start_python_time
        python_time_label.config(text=f'Час виконання: {round(python_run_time, 4)}
сек')
        python_accuracy_label.config(text=f'Похибка:
{"{:.3e}".format(python_accuracy)}')
        python_result_label.config(text=f'Результат: \n{python_answer}')
        # - - - - - #
        start_octave_time = time.time()
        octave_answer, octave_accuracy = run_octave()
        octave_run_time = time.time() - start_octave_time
        octave_time_label.config(text=f'Час виконання: {round(octave_run_time, 2)}
сек')
        octave_accuracy_label.config(text=f'Похибка:
{"{:.3e}".format(octave_accuracy)}')
        octave_result_label.config(text=f'Результат: \n{octave_answer}')
        # - - - - - #
        matrix_A_label.config(text=f'Матриця A:\n {matrix_A}')
        vector_b_label.config(text=f'Вектор b:\n {vector_b}')
    except:
        pass

python_name_label = tk.Label(root,
                        text='Python',
                        bg=CL_BLUE,
                        fg=CL_YELLOW,
                        font=('Helvetica', 20),
                        ).place(x=150, y=30)

python_time_label = tk.Label(root,
                        text='Час виконання: ',
                        bg=CL_BLUE,
                        fg=CL_YELLOW,
```

```
                        font=('Helvetica', 12),
                        )
python_time_label.place(x=100, y=80)

python_accuracy_label = tk.Label(root,
                        text='Похибка: ',
                        bg=CL_BLUE,
                        fg=CL_YELLOW,
                        font=('Helvetica', 12),
                        )
python_accuracy_label.place(x=100, y=110)

python_result_label = tk.Label(root,
                        text='Результат: ',
                        bg=CL_BLUE,
                        fg=CL_YELLOW,
                        font=('Helvetica', 12),
                        )
python_result_label.place(x=100, y=140)

# ---------------------------------------------- #

octave_name_label = tk.Label(root,
                        text='Octave',
                        bg=CL_BLUE,
                        fg=CL_GREEN,
                        font=('Helvetica', 20),
                        ).place(x=540, y=30)

octave_time_label = tk.Label(root,
                        text='Час виконання: ',
                        bg=CL_BLUE,
                        fg=CL_GREEN,
                        font=('Helvetica', 12),
                        )
octave_time_label.place(x=490, y=80)

octave_accuracy_label = tk.Label(root,
                        text='Похибка: ',
                        bg=CL_BLUE,
                        fg=CL_GREEN,
                        font=('Helvetica', 12),
                        )
octave_accuracy_label.place(x=490, y=110)

octave_result_label = tk.Label(root,
                        text='Результат: ',
                        bg=CL_BLUE,
                        fg=CL_GREEN,
                        font=('Helvetica', 12),
                        )
octave_result_label.place(x=490, y=140)
```

```
# ---------------------------------------------- #

matrix_A_label = tk.Label(root,
                          text='Матриця A: ',
                          bg=CL_BLUE,
                          fg=CL_RED,
                          font=('Helvetica', 12),
                          )
matrix_A_label.place(x=100, y=350)

vector_b_label = tk.Label(root,
                          text='Вектор b: ',
                          bg=CL_BLUE,
                          fg=CL_RED,
                          font=('Helvetica', 12),
                          )
vector_b_label.place(x=490, y=350)

# ---------------------------------------------- #

button_run = tk.Button(root,
                       text='Run',
                       command=run,
                       bg=CL_RED,
                       fg=CL_YELLOW,
                       borderwidth=0,
                       font=('Helvetica', 20),
                       ).place(x=360, y=30)
```

Вміст файлу python_.py :

```
import numpy as np
import tkinter.messagebox as messagebox


def run_python():
    def read_matrix(filename):
        """Функція для зчитування матриці з файлу
        """
        try:
            with open(filename, 'r') as file:
                lines = file.readlines()
                matrix = []
                for line in lines:
                    row = [float(x) for x in line.strip().split()]
                    matrix.append(row)
                return np.array(matrix)
        except FileNotFoundError:
            messagebox.showerror('ERROR', f'Помилка: Файл `{filename}` не
знайдено.')
        except ValueError:
```

```
            messagebox.showerror('ERROR', f'Помилка: Матриця в файлі `{filename}` містить неправильні значення.')
        except Exception as e:
            messagebox.showerror('ERROR', f'Помилка: Виникла невідома помилка при зчитуванні матриці: {str(e)}')

    def read_vector(filename):
        """Функція для зчитування вектора з файлу
        """
        try:
            with open(filename, 'r') as file:
                lines = file.readlines()
                vector = [float(line.strip()) for line in lines]
                return np.array(vector)
        except FileNotFoundError:
            messagebox.showerror('ERROR', f'Помилка: Файл `{filename}` не знайдено.')
        except ValueError:
            messagebox.showerror('ERROR', f'Помилка: Вектор в файлі `{filename}` містить неправильні значення.')
        except Exception as e:
            messagebox.showerror('ERROR', f'Помилка: Виникла невідома помилка при зчитуванні вектора: {str(e)}')


    # Зчитування матриці A та вектора b з файлів
    matrix_A = read_matrix('Labs\Lab_1\Matrix_A.txt')
    vector_b = read_vector('Labs\Lab_1\Vector_b.txt')

    if len(vector_b) != len(matrix_A):
        messagebox.showerror('ERROR', f'Помилка: Вектор b не відповідає розмірності матриці A.')

    # Розв'язання СЛАР Ax = b
    try:
        solution = np.linalg.solve(matrix_A, vector_b)

        # Обчислення похибки
        accuracy = np.mean((vector_b - np.dot(matrix_A, solution)) ** 2)
        # print(accuracy)
        # accuracy = np.max(np.abs(vector_b - np.dot(matrix_A, solution)))
        # print(accuracy)
        return [matrix_A, vector_b, solution, accuracy]
    except:
        pass


if __name__ == '__main__':
    a, b, x, e = run_python()
    print(f'Matrix:\n{a}')
    print(f'Vector: {b}')
    print(f'Solution: {x}')
    print(f'Accuracy: {e}')
```

Вміст файлу octave_.py :

```python
from oct2py import octave


def run_octave():
    """Використання Octave для завантаження даних та розв'язання системи рівнянь
    """
    octave.eval("A = load('Labs\Lab_1\Matrix_A.txt');")
    octave.eval("b = load('Labs\Lab_1\Vector_b.txt');")
    octave.eval("x = A \\ b;")

    # Отримання результату з Octave
    solution = octave.pull('x')
    solution = solution.flatten()

    # Обчислення похибки
    octave.eval("accuracy = norm(A * x - b);")
    accuracy = octave.pull('accuracy')

    return [solution, accuracy]

if __name__ == '__main__':
    x, e = run_octave()
    print(f'Solution: {x}')
    print(f'Accuracy: {e}')
```