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
Іваник Ю.П.

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 7

Матриця A				Вектор
3,1	1,5	1,8	1	10,83
1,5	2,5	0,5	1	9,20
1,0	0,5	4,2	1	17,10
1	1	1	1	1

Вимоги до ПЗ

1. Реалізувати перевірки на некоректний ввід (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

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 [1] 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 — 0.032 секунд
- Похибка — 1.775e-30
- Результат — [0.58229106   7.33150205   6.17679719 -13.0905903]

На мові Octave:

- Час виконання — 2.99 секунд
- Похибка — 2.665e-15
- Результат — [0.58229106   7.33150205   6.17679719 -13.0905903]

Така мала похибка може бути пов'язана з доцільним використанням LU- методу для заданої матриці.

Скріншоти виконання програми наведені в Додатку А.

Відповіді на контрольні питання

1. **У якому випадку система лінійних алгебраїчних рівнянь має єдиний розв'язок?** — Система лінійних алгебраїчних рівнянь має єдиний розв’язок, якщо кількість рівнянь співпадає з кількістю невідомих. Це також відбувається, коли детермінант (визначник) матриці не дорівнює нулю.
2. **Як використовується факторизація матриць для розв'язання систем лінійних рівнянь ?** — Факторизація матриць використовується для розв’язання систем лінійних рівнянь шляхом перетворення вихідної матриці системи на трикутний вигляд за допомогою лінійних перетворень.
3. **Охарактеризуйте прямі й ітераційні методи розв'язання систем лінійних рівнянь.** — Методи розв’язання систем лінійних алгебраїчних рівнянь поділяються на два типи: прямі (або точні) та ітераційні (або наближені). Прямий метод, такий як метод Гаусса, дає точний розв’язок за скінчену кількість кроків. Ітераційний метод, такий як метод Зейделя, навпаки, дозволяє отримати розв’язок лише з зазначеною точністю.
4. **Опишіть компактну схему методу Гауса.** — Компактна схема методу Гаусса - це економний спосіб запису обчислень, який повністю відповідає традиційному методу Гаусса.
5. **Опишіть алгоритм ітераційного методу Зейделя.** — Метод Зейделя - це модифікація методу простих ітерацій. Він полягає в обчисленнях **i**-ої координати нової точки x за допомогою відомих (**i-1**) координат нової точки та (**n-i+1**) координат старої точки.
6. **Дайте визначення збіжності ітераційного процесу.** — Збіжність ітераційного процесу означає, що процес ітерацій продовжується до досягнення необхідної точності розв’язку.
7. **Які методи називають самовиправляючимися?** — Самовиправляючі методи використовуються для зменшення впливу помилок, які можуть виникнути при обчисленнях, і для покращення точності результатів. Основні методи автоматичного виправлення помилок включають в себе:
    - Метод найменших квадратів (МНК): Цей метод дозволяє знайти найкращий наближений розв'язок СЛАР, навіть якщо вихідні дані містять помилки або шум. Він зменшує суму квадратів відхилень між реальними значеннями і передбаченими значеннями, щоб знайти оптимальні коефіцієнти розв'язку.
    - Метод QR-розкладу: Цей метод дозволяє розкласти матрицю системи на добуток ортогональної матриці і верхньотрикутної матриці. Це спрощує розв'язання СЛАР, оскільки верхньотрикутна матриця може бути легко розв'язана навіть у випадку чисельних помилок.
    - Методи регуляризації: Ці методи використовуються для стабілізації обчислень і запобігання перенавчанню в разі надмірного обміну

даними. Найпоширеніші методи регуляризації включають L1-регуляризацію (Lasso) та L2-регуляризацію (Ridge).

- Методи ітераційного покращення: Ці методи включають ітераційні алгоритми, такі як метод Гаусса-Зейделя, які дозволяють поетапно покращувати розв'язок СЛАР. Вони можуть бути використані для автоматичного виправлення результатів, якщо вони не задовольняють точності.

Ці методи допомагають покращити стійкість і точність обчислень в системах лінійних алгебраїчних рівнянь в разі наявності помилок або неточностей в даних або обчисленнях.

Перелік посилань

1. LU Метод - https://www.geeksforgeeks.org/l-u-decomposition-system-linear-equations/

Додаток А – скріншоти роботи програми

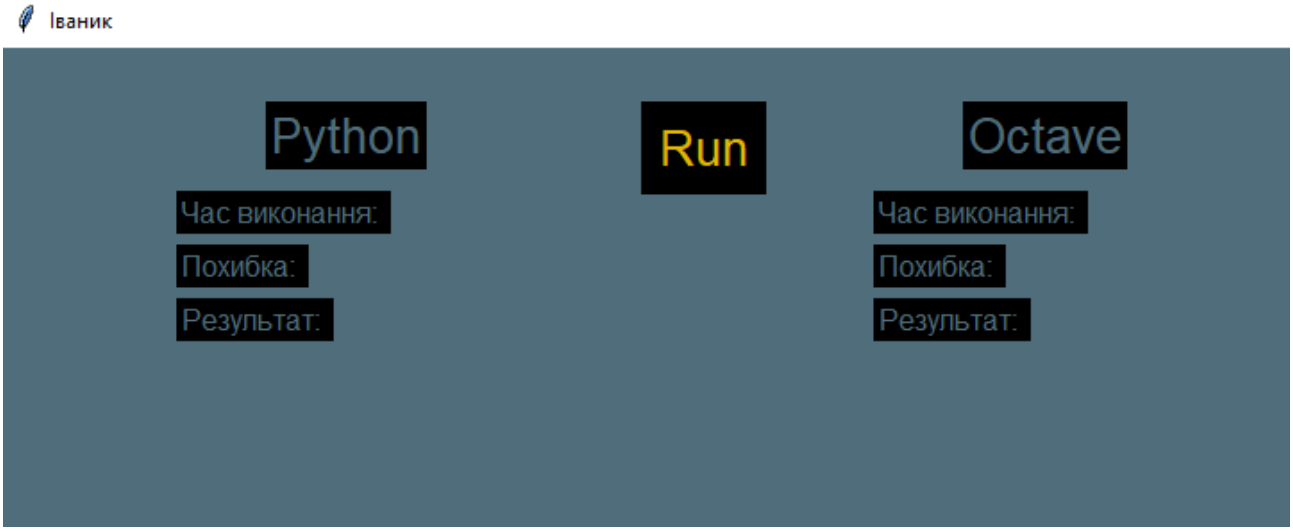

Рис. 1 – Вікно програми до запуску

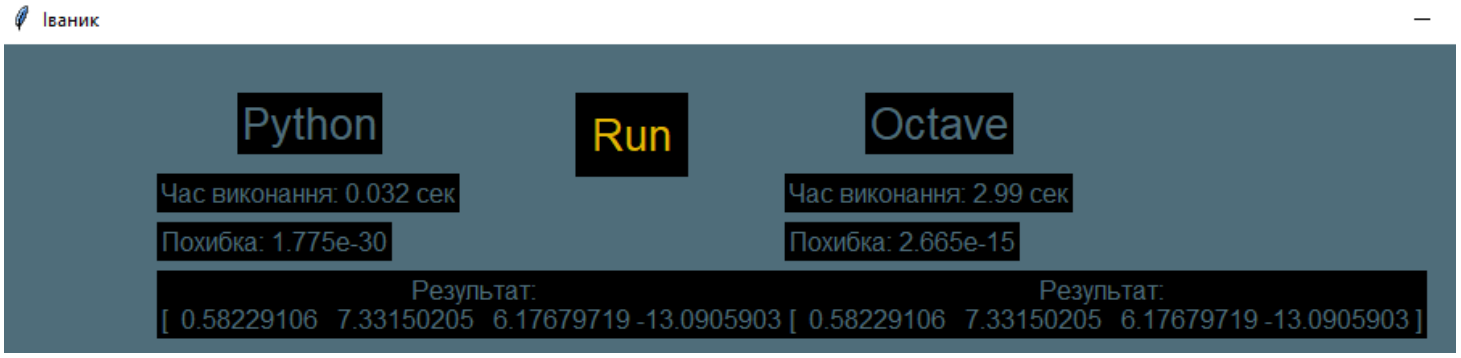

Рис. 2 – Вікно програми після запуску

Додаток Б – Блок-схема роботи програми

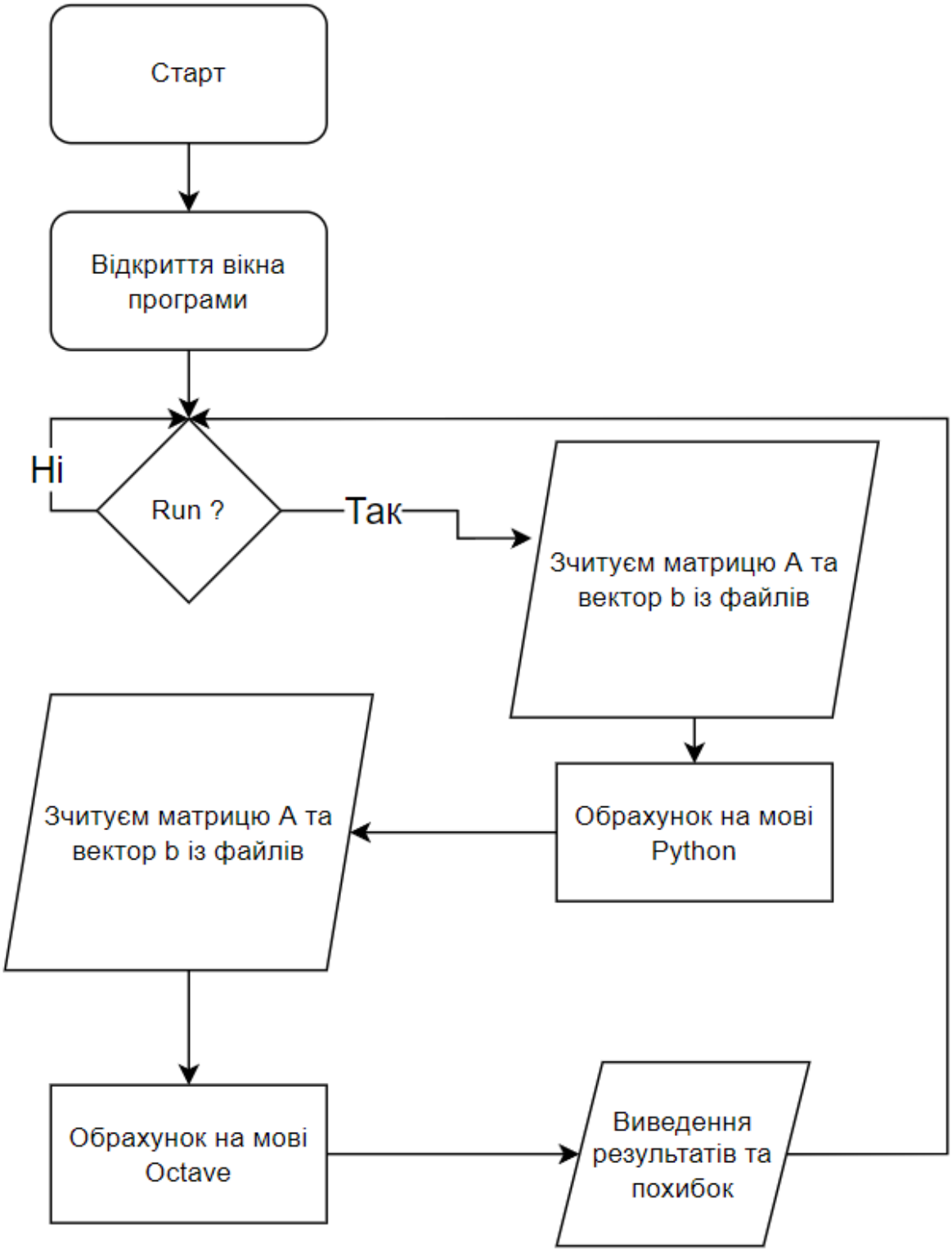

Рис. 3 – Блок-схема роботи програми

Додаток В – Код програми

Вміст файлу Matrix_A.txt :

3.1 1.5 1.8 1.0

1.5 2.5 0.5 1.0

1.0 0.5 4.2 1.0

1.0 1.0 1.0 1.0

Вміст файлу Vector_b.txt :

10.83

9.20

17.10

1

Вміст файлу main.py :

```
from window import *
import os


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

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'
CL_BLACK = '#000000'


root = tk.Tk()  #  Створення головного вікна програми
root.title('Іваник')  #  Задання назви вікну
root.geometry('1000x600')  #  Задаємо розмір вікна
root.resizable(width=False, height=False)  #  Заборона зміни розмірів вікна
root.configure(bg=CL_GRAY)  #  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}')
        print(python_answer)
        print(octave_answer)
    except:
        pass

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

python_time_label = tk.Label(root,
                        text='Час виконання: ',
                        bg=CL_BLACK,
                        fg=CL_GRAY,
                        font=('Helvetica', 12),
                        )
python_time_label.place(x=100, y=80)

python_accuracy_label = tk.Label(root,
```

```
                        text='Похибка: ',
                        bg=CL_BLACK,
                        fg=CL_GRAY,
                        font=('Helvetica', 12),
                        )
python_accuracy_label.place(x=100, y=110)

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

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

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

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

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

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


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

button_run = tk.Button(root,
                       text='Run',
```

```
                        command=run,
                        bg=CL_BLACK,
                        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\Ivanyk\Lab_1\Matrix_A.txt')
    vector_b = read_vector('Labs\Ivanyk\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 :

```
from oct2py import octave


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

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