Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра обчислювальної техніки

Лабораторна робота № 4

з дисципліни «Чисельні методи»

на тему

**“Обчислення власних значень та власних векторів матриць”**

Виконав:

студент гр. ІП-93

Завальнюк Максим

Викладач:

доц. Рибачук Л.В.

Київ – 2021

### Зміст

[Зміст 2](#_Toc67845271)

[1 Постановка задачі 3](#_Toc67845272)

[2 Розв’язок 4](#_Toc67845273)

[3 Розв’язок за допомогою NumPy 6](#_Toc67845274)

[4 Лістинг програми 7](#_Toc67845275)

### 1 Постановка задачі

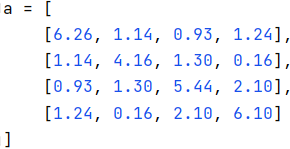
Створити програму, для приведення матриці А до нормальної форми Фробеніюса.

Отримане характеристичне рівняння розв’язати довільним способом у NumPy і отримати всі власні числа λі, і = 1,...,m з точністю 5 знаків після коми. Знайти по одному власному вектору для кожного власного числа.

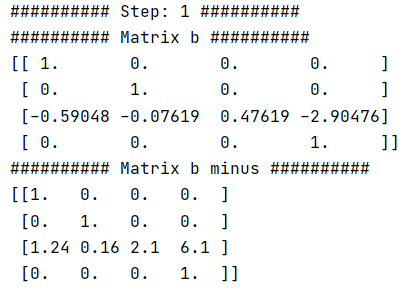
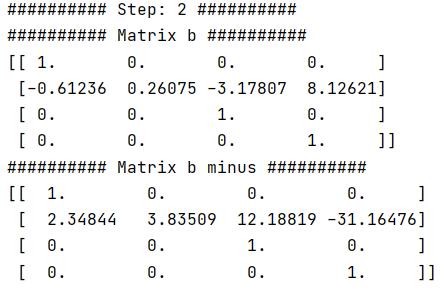
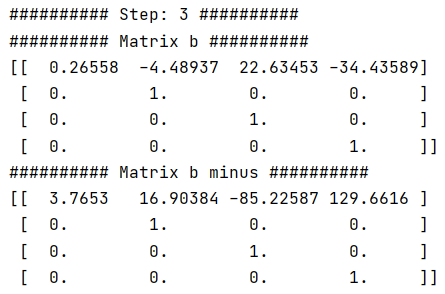
Перевірити точність знайдених результатів, підставляючи у рівняння Ax = λx(1) знайдені власні числа та власні вектори. Знайти власні числа матриці А виключно за допомогою NumPy і порівняти з отриманими раніше результатами.

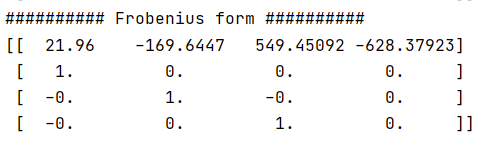
### 2 Розв’язок

Коефіцієнти: a=0.11 \* 0 = 0; b = 0.02 \* (-2) = -0.04; g = 0.02 \* (-2) = -0.04; d = 0.015 \* 0 = 0.

Початкова матриця системи А

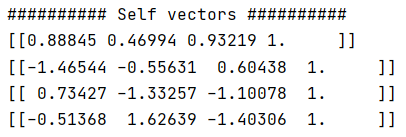
Нижче приведені результати виконання програми.

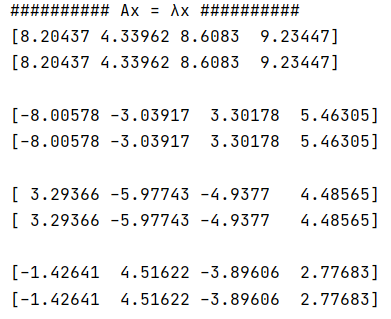
Проміжні матриці та :

Результуюча матриця Р у нормальній формі Фробеніюса:

Отримане характеристичне рівняння:

Власні числа – корені характеристичного рівняння:

Власний вектор для кожного власного числа:

Оцінка точності обчислень (підстановка результатів у вихідне рівняння (1)):

### 3 Розв’язок за допомогою NumPy

Нижче наведено розв’язок системи у NumPy:

v, w = np.linalg.eigh(matrix\_a)  
print(template.substitute(string=**'NumPy numbers'**))  
print(v.round(5))

Порівняння отриманого результату (п. 2) із результатом з NumPy за допомогою методу середньоквадратичної похибки:

print(template.substitute(string=**'Fault'**))  
print(round(fault.get\_fault(sorted(self\_numbers), sorted(v)), 5))

Це дуже добре, оскільки результат вважають гарним, якщо відносна похибка не перевищує 0.1 %. У даному випадку взагалі 0 %.

### 4 Лістинг програми

from string import Template  
from checker import fault  
  
import numpy as np  
from typing import Tuple  
  
template = Template(**'#'** \* 10 + **' $string '** + **'#'** \* 10)  
np.set\_printoptions(suppress=True)  
  
a = [  
 [6.26, 1.14, 0.93, 1.24],  
 [1.14, 4.16, 1.30, 0.16],  
 [0.93, 1.30, 5.44, 2.10],  
 [1.24, 0.16, 2.10, 6.10]  
]  
  
  
def frobenius(matrix: list) -> Tuple[np.matrix, np.matrix]:  
 *"""  
 Get a Frobenius form  
 :param matrix: start matrix  
 :return: normalised matrix and S matrix for future  
 """* length = len(matrix)  
 s\_matrix = np.identity(length)  
 for i in range(length - 1, 0, -1):  
 matrix\_b = np.identity(length)  
 matrix\_b\_minus = matrix\_b.copy()  
  
 *# Fill matrix b and minus one b* for j in range(length):  
 if j == i - 1:  
 matrix\_b[i - 1][j] = 1 / matrix[i][i - 1]  
 else:  
 matrix\_b[i - 1][j] = matrix[i][j] / matrix[i][i - 1] \* (-1)  
 matrix\_b\_minus[i - 1][j] = matrix[i][j]  
 print(template.substitute(string=**f'Step:** {abs(i - length)}**'**))  
 print(template.substitute(string=**'Matrix b'**))  
 print(matrix\_b.round(5))  
 s\_matrix = np.dot(s\_matrix, matrix\_b)  
 print(template.substitute(string=**'Matrix b minus'**))  
 print(matrix\_b\_minus.round(5))  
 matrix = np.dot(matrix\_b\_minus, np.dot(matrix, matrix\_b))  
 print(template.substitute(string=**'Temporary result'**))  
 print(matrix.round(5))  
 return matrix, s\_matrix  
  
  
def get\_self\_numbers(coefficients: list) -> np.matrix:  
 *"""  
 Function to get self numbers  
 :param coefficients: numbers from the first raw in normalised matrix  
 :return: self numbers of matrix  
 """* coefficients = list(coefficients)  
 coefficients = [round(coef \* (-1), 5) for coef in coefficients]  
 coefficients.insert(0, 1)  
 *# Print equation area start* equation = **''** for index in range(len(coefficients)):  
 equation += **'+('** + str(coefficients[index]) + **'\*λ^'** + str(abs(index - len(coefficients) + 1)) + **') '** equation += **'= 0'** print(template.substitute(string=**'Characteristic equation'**))  
 print(equation)  
 *# Print equation area end* roots = np.roots(coefficients)  
 return roots  
  
  
def get\_self\_vectors(self\_numbers: list, s\_matrix: np.matrix) -> list:  
 *"""  
 Function to get self vectors  
 :param self\_numbers: self numbers of matrix  
 :param s\_matrix: S matrix from earlier Frobenius function  
 :return: self vectors  
 """* self\_vectors = []  
 y\_array = []  
 for number in self\_numbers:  
 temp\_array = []  
 for i in range(len(self\_numbers)):  
 temp\_array.insert(0, pow(number, i))  
 y\_array.append(temp\_array)  
 print(template.substitute(string=**'Y array'**))  
 print(np.matrix(y\_array).round(5))  
 print(template.substitute(string=**'S matrix'**))  
 print(s\_matrix.round(5))  
 print(template.substitute(string=**'Self vectors'**))  
 for element in y\_array:  
 vector = np.dot(s\_matrix, element)  
 self\_vectors.append(vector)  
 print(np.matrix(vector).round(5))  
 return self\_vectors  
  
  
def check\_vectors(matrix\_a: list, self\_numbers: list, self\_vectors: list) -> None:  
 *"""  
 Check the equation Ax = λx  
 :param matrix\_a: start matrix  
 :param self\_numbers: self numbers of matrix A  
 :param self\_vectors: self vectors of matrix A  
 :return: nothing to return  
 """* print(template.substitute(string=**'Ax = λx'**))  
 for index in range(len(self\_numbers)):  
 print(np.dot(matrix\_a, self\_vectors[index]).round(5))  
 print(np.dot(self\_numbers[index], self\_vectors[index]).round(5))  
 print()  
  
  
def main\_part(matrix\_a: list) -> None:  
 *"""  
 Get all functions in one place  
 :param matrix\_a: start matrix  
 :return: nothing to return  
 """* normal\_form, s\_matrix = frobenius(matrix\_a)  
 print(template.substitute(string=**'Frobenius form'**))  
 print(normal\_form.round(5))  
 self\_numbers = get\_self\_numbers(normal\_form[0])  
 print(template.substitute(string=**'Self numbers'**))  
 print(self\_numbers.round(5))  
 v, w = np.linalg.eigh(matrix\_a)  
 print(template.substitute(string=**'NumPy numbers'**))  
 print(v.round(5))  
 self\_vectors = get\_self\_vectors(self\_numbers, s\_matrix)  
 check\_vectors(matrix\_a, self\_numbers, self\_vectors)  
 print(template.substitute(string=**'Fault'**))  
 print(round(fault.get\_fault(sorted(self\_numbers), sorted(v)), 5))  
  
  
print(**'Matrix A:'**)  
print(np.matrix(a))  
main\_part(a.copy())