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

1 Постановка задачі

Створити програму, для приведення матриці A до нормальної форми Фробеніюса. Отримане характеристичне рівняння розв’язати довільним способом у NumPy і отримати всі власні числа λi, $i = 1,...,m$ з точністю 5 знаків після коми. Знайти по одному власному вектору для кожного власного числа.

Перевірити точність знайдених результатів, підставляючи у рівняння $Ax = \lambda x$(1) знайдені власні числа та власні вектори. Знайти власні числа матриці A 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 A

```
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]
]
```

Нижче приведені результати виконання програми.

Проміжні матриці M_i^{-1} та M_i:

```
########## Step: 1 ##########
########## Matrix b ##########
[[ 1.       0.       0.       0.     ]
 [ 0.       1.       0.       0.     ]
 [-0.59048 -0.07619  0.47619 -2.90476]
 [ 0.       0.       0.       1.     ]]
########## Matrix b minus ##########
[[1.   0.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [1.24 0.16 2.1  6.1 ]
 [0.   0.   0.   1.  ]]
########## Step: 2 ##########
########## Matrix b ##########
[[ 1.       0.       0.       0.     ]
 [-0.61236  0.26075 -3.17807  8.12621]
 [ 0.       0.       1.       0.     ]
 [ 0.       0.       0.       1.     ]]
########## Matrix b minus ##########
[[  1.        0.        0.        0.     ]
 [  2.34844   3.83509  12.18819 -31.16476]
 [  0.        0.        1.        0.     ]
 [  0.        0.        0.        1.     ]]
########## Step: 3 ##########
########## Matrix b ##########
[[  0.26558  -4.48937  22.63453 -34.43589]
 [  0.        1.        0.        0.     ]
 [  0.        0.        1.        0.     ]
 [  0.        0.        0.        1.     ]]
########## Matrix b minus ##########
[[  3.7653   16.90384 -85.22587 129.6616 ]
 [  0.        1.        0.        0.     ]
 [  0.        0.        1.        0.     ]
 [  0.        0.        0.        1.     ]]
```

Результуюча матриця P у нормальній формі Фробеніюса:

```
########## Frobenius form ##########
[[  21.96    -169.6447   549.45092 -628.37923]
 [   1.         0.         0.         0.     ]
 [  -0.         1.        -0.         0.     ]
 [  -0.         0.         1.         0.     ]]
```

Отримане характеристичне рівняння:

```
########## Characteristic equation ##########
+(1*λ^4) +(-21.96*λ^3) +(169.6447*λ^2) +(-549.45092*λ^1) +(628.37923*λ^0) = 0
```

Власні числа – корені характеристичного рівняння:

```
########## Self numbers ##########
[9.23447 5.46305 4.48565 2.77683]
```

Власний вектор для кожного власного числа:

```
########## Self vectors ##########
[[0.88845 0.46994 0.93219 1.     ]]
[[-1.46544 -0.55631  0.60438  1.     ]]
[[ 0.73427 -1.33257 -1.10078  1.     ]]
[[-0.51368  1.62639 -1.40306  1.     ]]
```

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

```
########## Ax = λx ##########
[8.20437 4.33962 8.6083  9.23447]
[8.20437 4.33962 8.6083  9.23447]

[-8.00578 -3.03917  3.30178  5.46305]
[-8.00578 -3.03917  3.30178  5.46305]

[ 3.29366 -5.97743 -4.9377   4.48565]
[ 3.29366 -5.97743 -4.9377   4.48565]

[-1.42641  4.51622 -3.89606  2.77683]
[-1.42641  4.51622 -3.89606  2.77683]
```

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))
```

```
########## NumPy numbers ##########
[2.77683 4.48565 5.46305 9.23447]
```

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))
```

```
########## Fault ##########
0.0
```

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())
```