Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Методы численного анализа»

ОТЧЕТ

к лабораторной работе №5

на тему:

«ВЫЧИСЛЕНИЕ СОБСТВЕННЫХ ЗНАЧЕНИЙ И ВЕКТОРОВ»

БГУИР 6-05 0612 02 86

Выполнил студент группы 353505
МАРТЫНКЕВИЧ Евгений Дмитриевич

(дата, подпись студента)

Проверил доцент кафедры информатики
АНИСИМОВ Владимир Яковлевич

(дата, подпись преподавателя)

Минск 2024

Содержание

Цель работы

- изучить метод Якоби (вращений) для поиска собственных значений и векторов матрицы;
- составить программу поиска собственных значений и векторов матрицы указанными методами, применимую для организации вычислений на ЭВМ;
- выполнить тестовые примеры и проверить правильность работы программы

ЗАДАНИЕ 5. С точностью 0,0001 вычислить собственные значения и собственные векторы матрицы А ,

где A = kC + D, A – исходная матрица для расчёта, k – номер варианта (0-15), матрицы C, D заданы ниже:

$$C = \begin{bmatrix} 0,2 & 0 & 0,2 & 0 & 0 \\ 0 & 0,2 & 0 & 0,2 & 0 \\ 0,2 & 0 & 0,2 & 0 & 0,2 \\ 0 & 0,2 & 0 & 0,2 & 0 \\ 0 & 0 & 0,2 & 0 & 0,2 \end{bmatrix}, \quad D = \begin{bmatrix} 2,33 & 0,81 & 0,67 & 0,92 & -0,53 \\ 0,81 & 2,33 & 0,81 & 0,67 & 0,92 \\ 0,67 & 0,81 & 2,33 & 0,81 & 0,92 \\ 0,92 & 0,67 & 0,81 & 2,33 & -0,53 \\ -0,53 & 0,92 & 0,92 & -0,53 & 2,33 \end{bmatrix}.$$

Вариант 3

Программная реализация

Исходные данные

Матрица, полученная в результате подстановки A = 3 * C + D:

```
Исходная матрица:
[[ 2.93  0.81  1.27  0.92 -0.53]
 [ 0.81  2.93  0.81  1.27  0.92]
 [ 1.27  0.81  2.93  0.81  1.52]
 [ 0.92  1.27  0.81  2.93 -0.53]
 [-0.53  0.92  1.52 -0.53  2.93]]
```

Код метода Якоби(вращений):

```
def jacobi_method(matrix):
    eigenvectors = numpy.eye(len(matrix))
    temp_A = matrix.copy()
    count = 0;
    while(not_diag_squares(temp_A) > 1e-4):
        temp_V = numpy.eye(len(temp_A))
        m_i, m_j = not_diag_max_element(temp_A)
        if(temp_A[m_i][m_i] - temp_A[m_j][m_j] < 1e-10):
            cos_phi = 1 / (2 ** (1/2))
            if(temp_A[m_i][m_j] > 0):
                sin_phi = 1 / (2 ** (1/2))
            else:
                sin_phi = -1 / (2 ** (1/2))
        else:
            cos_phi = math.cos(0.5 * math.atan(2*temp_A[m_i][m_j] /
(temp_A[m_i][m_i] - temp_A[m_j][m_j])))
            sin_phi = math.sin(0.5 * math.atan(2*temp_A[m_i][m_j] /
(temp_A[m_i][m_i] - temp_A[m_j][m_j])))
        temp_V[m_i][m_i] = cos_phi
        temp_V[m_j][m_j] = cos_phi
        temp_V[m_i][m_j] = sin_phi * -1
        temp_V[m_j][m_i] = sin_phi
        temp_A = numpy.dot(numpy.dot(temp_V.transpose(), temp_A),
temp_V)
       eigenvectors = numpy.dot(eigenvectors, temp_V)
       count += 1
    print(f"Кол-во итераций: {count}")
    return temp_A, eigenvectors
```

```python
def check_matrix_simmetrix(matrix):
    if(not numpy.all(matrix == matrix.transpose())):
        print("Матрица не является симметричной!")
        exit()


def not_diag_max_element(matrix):
    max = matrix[0][1]
    max_i = 0
    max_j = 1
    for i in range(len(matrix)):
        for j in range(len(matrix)):
            if(j > i):
                if(abs(matrix[i][j]) > abs(max)):
                    max = matrix[i][j]
                    max_i = i
                    max_j = j
    return max_i, max_j


def not_diag_squares(matrix):
    sum = 0.0
    for i in range(len(matrix)):
        for j in range(len(matrix)):
            if(j > i):
                sum += 2 * matrix[i][j] ** 2
    return sum
```

Полученные результаты

```
Исходная матрица:
[[ 2.93  0.81  1.27  0.92 -0.53]
 [ 0.81  2.93  0.81  1.27  0.92]
 [ 1.27  0.81  2.93  0.81  1.52]
 [ 0.92  1.27  0.81  2.93 -0.53]
 [-0.53  0.92  1.52 -0.53  2.93]]
Кол-во итераций: 21
Матрица собственных значений(после метода):
[ 6.0629e+00 -3.3557e-04 -7.6058e-08 -1.7347e-18 -3.7696e-06 ]
[ -3.3557e-04 4.0841e+00 7.6283e-07 2.3236e-03 1.6759e-06 ]
[ -7.6058e-08 7.6283e-07 2.5048e+00 -5.0383e-05 -1.8540e-04 ]
[ -2.6368e-16 2.3236e-03 -5.0383e-05 1.6046e+00 -1.2994e-03 ]
[ -3.7696e-06 1.6759e-06 -1.8540e-04 -1.2994e-03 3.9352e-01 ]
Собственные значения(диагональ):
[6.06292661 4.08407439 2.50483255 1.60464996 0.39351649]
Матрица свободных векторов:
[[ 0.43330067 -0.39265725  0.56083885 -0.44296328  0.38381522]
 [ 0.50629106  0.02410453 -0.58452407 -0.54546625 -0.32231431]
 [ 0.54774965  0.26723712  0.43477908  0.37667817 -0.54555966]
 [ 0.42859056 -0.44775334 -0.38663034  0.60351409  0.31955408]
 [ 0.26870122  0.75719253 -0.0726328   0.01159352  0.59080386]]
Матрица после операции V * A * V.T:
[[ 2.93  0.81  1.27  0.92 -0.53]
 [ 0.81  2.93  0.81  1.27  0.92]
 [ 1.27  0.81  2.93  0.81  1.52]
 [ 0.92  1.27  0.81  2.93 -0.53]
 [-0.53  0.92  1.52 -0.53  2.93]]
Определители матрицы для каждого соб. знач.
[ 1.0127e-05 -6.0645e-05 2.0384e-07 9.4762e-06 -7.5461e-05 ]
```

По значениям определителей матрицы для каждого собственного значения можно сделать вывод, что точность 0,0001 была достигнута.

Тестовый пример 1. Нулевые строка и столбец

```
Исходная матрица:
[[0. 0. 0.]
 [0. 7. 3.]
 [0. 3. 4.]]
Кол-во итераций: 1
Матрица собственных значений(после метода):
[ 0.0000e+00 0.0000e+00 0.0000e+00 ]
[ 0.0000e+00 8.8541e+00 4.4409e-16 ]
[ 0.0000e+00 2.2204e-16 2.1459e+00 ]
Собственные значения(диагональ):
[0.         8.85410197 2.14589803]
Матрица свободных векторов:
[[ 1.          0.          0.        ]
 [ 0.          0.85065081 -0.52573111]
 [ 0.          0.52573111  0.85065081]]
Матрица после операции V * A * V.T:
[[0. 0. 0.]
 [0. 7. 3.]
 [0. 3. 4.]]
Определители матрицы для каждого соб. знач.
[ 0.0000e+00 -1.1796e-14 0.0000e+00 ]
```

Тестовый пример 2. Нулевая диагональ

```
Исходная матрица:
[[ 0.  3.  7.]
 [ 3.  0. 11.]
 [ 7. 11.  0.]]
Кол-во итераций: 7
Матрица собственных значений(после метода):
[ 1.4520e+01 -6.1972e-07 -3.4557e-04 ]
[ -6.1972e-07 -2.6897e+00 -4.3368e-18 ]
[ -3.4557e-04 -1.1796e-16 -1.1830e+01 ]
Собственные значения(диагональ):
[ 14.51960878  -2.6897146  -11.82989417]
Матрица свободных векторов:
[[ 0.44505376 -0.85044437 -0.28048444]
 [ 0.59736648  0.51528705 -0.61451814]
 [ 0.66714349  0.10594161  0.7373574 ]]
Матрица после операции V * A * V.T:
[[-1.99840144e-15  3.00000000e+00  7.00000000e+00]
 [ 3.00000000e+00 -2.66453526e-15  1.10000000e+01]
 [ 7.00000000e+00  1.10000000e+01 -1.77635684e-15]]
Определители матрицы для каждого соб. знач.
[ 2.0551e-06 3.5748e-12 -1.0915e-06 ]
```

Тестовый пример 3. Нулевая матрица

```
Исходная матрица:
[[3. 3. 3.]
 [3. 3. 3.]
 [3. 3. 3.]]
Кол-во итераций: 2
Матрица собственных значений(после метода):
[ 9.0000e+00 0.0000e+00 8.8818e-16 ]
[ 0.0000e+00 0.0000e+00 0.0000e+00 ]
[ 3.6260e-16 0.0000e+00 -2.5640e-16 ]
Собственные значения(диагональ):
[ 9.00000000e+00  0.00000000e+00 -2.56395025e-16]
Матрица свободных векторов:
[[ 0.57735027 -0.70710678 -0.40824829]
 [ 0.57735027  0.70710678 -0.40824829]
 [ 0.57735027  0.          0.81649658]]
Матрица после операции V * A * V.T:
[[3. 3. 3.]
 [3. 3. 3.]
 [3. 3. 3.]]
Определители матрицы для каждого соб. знач.
[ 0.0000e+00 0.0000e+00 3.5499e-30 ]
```

Тестовый пример 3. Несимметричная матрица

```
Исходная матрица:
[[9. 8. 7.]
 [4. 5. 6.]
 [3. 2. 1.]]
Матрица не является симметричной!
```

Тестовый пример 3. Несимметричная матрица(без проверки)

```
Исходная матрица:
[[9. 8. 7.]
 [4. 5. 6.]
 [3. 2. 1.]]
Кол-во итераций: 9
Матрица собственных значений(после метода):
[ 1.5390e+01 4.3713e-04 5.5231e-03 ]
[ -6.4548e+00 1.5313e-02 4.0828e-04 ]
[ -1.0052e+00 -2.3035e+00 -4.0538e-01 ]
Собственные значения(диагональ):
[ 1.53900638e+01  1.53130902e-02 -4.05376860e-01]
Матрица свободных векторов:
[[ 0.61388435 -0.73603235  0.28531102]
 [ 0.57682284  0.17150817 -0.7986616 ]
 [ 0.53890761  0.65485977  0.52984646]]
Матрица после операции V * A * V.T:
[[9. 8. 7.]
 [4. 5. 6.]
 [3. 2. 1.]]
Определители матрицы для каждого соб. знач.
[ -4.7806e-02 9.5392e-02 9.9311e-02 ]
```

Вывод

В ходе выполнения лабораторной работы я изучил метод Якоби(вращений), написал программу его реализации на языке Python, правильность работы программы проверил на тестовых примерах.

На основании тестов можно сделать следующие выводы:

- Программа позволяет получить решения системы с заданной точностью (заданная точность в условиях лабораторной работы 10^-4);
- Основное достоинство метода Якоби заключается в том, что при выполнении каждого плоского вращения уменьшается сумма квадратов недиагональных элементов; сходимость этой суммы к нулю по мере увеличения числа шагов гарантирует сходимость процесса диагонализации.