Министерство образования Республики Беларусь
Учреждение образования
“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы численного анализа

ОТЧЁТ
к лабораторной работе 5
на тему

Вычисление собственных значений и векторов

Выполнил: студент группы 053502
Герчик Артём Вадимович

Проверил: Анисимов Владимир Яковлевич

Минск 2022

Оглавление

Цели выполнения задания

1) Освоить методы вычисления собственных значений и векторов

Краткие теоретические сведения

Итеративные алгоритмы решают задачу вычисления собственных значений путём построения последовательностей, сходящихся к собственным значениям. Некоторые алгоритмы дают также последовательности векторов, сходящихся к собственным векторам. Чаще всего последовательности собственных значений выражаются через последовательности подобных матриц, которые сходятся к треугольной или диагональной форме, что позволяет затем просто получить собственные значения. Последовательности собственных векторов выражаются через соответствующие матрицы подобия.

Метод Якоби (вращений) использует итерационный процесс, который приводит исходную симметрическую матрицу A к диагональному виду с помощью последовательности *элементарных ортогональных преобразований* (в дальнейшем называемых *вращениями Якоби* или *плоскими вращениями).* Процедура построена таким образом, что на $(k+1)$-ом шаге осуществляется преобразование вида

$$A^{(k)} \to A^{(k+1)} = V^{(k)*} A^{(k)} V^{(k)} = V^{(k)*} \dots V^{(0)*} A^{(0)} V^{(0)} \dots V^{(k)}, \quad k=0,1,2\dots, \qquad (5.1)$$

где $A^{(0)} = A$, $V^{(k)} = V^{(k)}_{ij}(\varphi)$ — ортогональная матрица, отличающаяся от единичной матрицы только элементами

$$v_{ii} = v_{jj} = \cos\varphi, \quad v_{ij} = -v_{ji} = -\sin\varphi, \qquad (5.2)$$

значение φ выбирается при этом таким образом, чтобы обратить в 0 наибольший по модулю недиагональный элемент матрицы $A^{(k)}$. Итерационный процесс постепенно приводит к матрице со значениями недиагональных элементов, которыми можно пренебречь, т.е. матрица $A^{(k)}$ все более похожа на диагональную, а диагональная матрица Λ является пределом последовательности $A^{(k)}$ при $k \to \infty$.

Алгоритм метода вращений.

1) В матрице $A^{(k)}$ $(k=0,1,2,\dots.)$ среди всех недиагональных элементов выбираем максимальный по абсолютной величине элемент, стоящий выше главной диагонали; определяем его номера i и j строки и столбца, в которых он стоит (если максимальных элементов несколько, можно взять любой из них);

2) По формулам

$$\cos\varphi_k = \cos\left(\frac{1}{2}\cdot \operatorname{arctg} P_k\right),$$

$$\sin\varphi_k = \sin\left(\frac{1}{2}\cdot \operatorname{arctg} P_k\right),$$

$$\text{где } P_k = \begin{cases} \dfrac{\pi}{4}, & \text{если } a_{ii}^{(k)} = a_{jj}^{(k)} \\ \dfrac{2\,a_{ij}^{(k)}}{a_{ii}^{(k)} - a_{jj}^{(k)}}, & \text{иначе} \end{cases}$$

вычисляем $\cos\varphi_k$ и $\sin\varphi_k$, получаем матрицу $V^{(k)} = V^{(k)}{}_{ij}(\varphi_k)$

$$V^{(k)} = \begin{array}{c} \\ \\ i \\ \\ \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccc} & & & & i & & & & j \end{array} \\ \begin{bmatrix} 1 & \dots & \dots & \dots & 0 & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \cos\varphi_k & \dots & \dots & \dots & -\sin\varphi_k & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & 1 & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \sin\varphi_k & \dots & \dots & \dots & \cos\varphi_k & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & 0 & \dots & \dots & \dots & 1 \end{bmatrix} \end{array}$$

3) По формуле

$$A^{(k+1)} = V^{(k)\mathrm{T}} \cdot A^{(k)} \cdot V^{(k)}$$

находим матрицу $A^{(k+1)}$.

4) Итерационный процесс останавливаем, когда в пределах принятой точности суммой квадратов всех недиагональных элементов матрицы $A^{(k+1)}$ можно пренебречь.

5) В качестве собственных значений матрицы A берем диагональные элементы матрицы $A^{(k+1)}$, в качестве собственных векторов – соответствующие столбцы матрицы

$$V = V^{(0)} V^{(1)} \dots V^{(k)}.$$

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.

Степенной метод или метод степенных итераций — итерационный алгоритм поиска собственного значения с максимальной абсолютной величиной и одного из соответствующих собственных векторов для произвольной матрицы.

Алгоритм прост и сходится со скоростью геометрической прогрессии если все максимальные по модулю собственные значения совпадают, в противном случае сходимости нет.

В начале алгоритма генерируется случайный вектор r_0. Далее проводятся последовательные вычисления по итеративной формуле:

$$r_{k+1} = \frac{Ar_k}{\|Ar_k\|}$$

Последовательность

$$\mu_k = \frac{r_k^T A r_k}{r_k^T r_k}$$

при указанном выше условии сходится к максимальному по модулю собственному значению, а вектор r_k образует соответствующий собственный вектор.

Задание

Вариант 7

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

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

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

Программная реализация

```
import numpy

print("Собственные вектора и собственные значения\n")

EPS = 10.0 ** -4

numpy.set_printoptions(suppress=True, precision=4,
floatmode="fixed")


def input_values():
    C = numpy.array([
        [0.2, 0.0, 0.2, 0.0, 0.0],
        [0.0, 0.2, 0.0, 0.2, 0.0],
        [0.2, 0.0, 0.2, 0.0, 0.2],
        [0.0, 0.2, 0.0, 0.2, 0.0],
        [0.0, 0.0, 0.2, 0.0, 0.2]
    ])
    D = numpy.array([
        [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]
    ])
    matrix_A = 7 * C + D

    return matrix_A


matrix_A = input_values()
n = len(matrix_A)
print(f'Матрица A:\n {matrix_A}\n')

if abs((matrix_A - matrix_A.T) ** 2).sum() > EPS:
    raise ValueError("Ошибка, матрица A неправильная")

count_of_iterations = 0

ans_matrix_V = numpy.eye(n)
```

```
while True:
    count_of_iterations += 1
    max_elem = (0, 1)
    for i in range(n):
        for j in range(i + 1, n):
            if abs(matrix_A[i][j]) >
abs(matrix_A[max_elem]):
                max_elem = (i, j)
    (i, j) = max_elem
    if matrix_A[i][i] == matrix_A[j][j]:
        p = numpy.pi / 4
    else:
        p = 2 * matrix_A[i][j] / (matrix_A[i][i] -
matrix_A[j][j])
    cos = numpy.cos(1 / 2 * numpy.arctan(p))
    sin = numpy.sin(1 / 2 * numpy.arctan(p))
    V = numpy.eye(n)

    V[i][i] = cos
    V[i][j] = -sin
    V[j][i] = sin
    V[j][j] = cos

    matrix_A = V.T @ matrix_A @ V  # matrix multiply
    ans_matrix_V = ans_matrix_V @ V  # matrix multiply

    if abs(matrix_A -
numpy.diag(numpy.diag(matrix_A))).sum() < EPS:
        ansW = numpy.diag(matrix_A)
        break


def normalization(W, V):
    V = numpy.array([(-i if i[0] < 0 else i) for i in
V.T]).T
    (W, V) = list(zip(*(sorted(list(zip(W, V.T)),
key=lambda t: t[0]))))
    W = numpy.array(W)
    V = numpy.array(V).T
    return (W, V)
```

```
matrix_A = input_values()

print('Метод вращений Якоби: ')
(W, V) = normalization(ansW, ans_matrix_V)
print(f"Собственные значения = {W}")
print(f"Собственные векторы = \n {V}")
print("Потребовалось итераций: =", count_of_iterations)

print("\nСтепенной метод:")
matrix_A = input_values()
r = numpy.ones(len(matrix_A))
count_of_iterations = 0
while True:
    count_of_iterations += 1
    old_u = (r.T @ matrix_A @ r) / (r.T @ r)
    r = (matrix_A @ r) / numpy.sqrt(sum((matrix_A @ r) **
2))
    u = (r.T @ matrix_A @ r) / (r.T @ r)
    if abs(u - old_u) < EPS:
        break
print(f'Максимальное, по модулю, собственное значение =
{u:.4f}')
print(f'Максимальный собственный вектор = {r}')
print("Количество итераций = ", count_of_iterations)
```

Полученные результаты

Тестовый пример 1

Вычислить с точностью 0.0001 собственные значения и собственные векторы матрицы **A** =

$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Ответ:

Тип алгоритма	Вращений Якоби	Степенной
Собственные значения:	$\lambda = [1.0000, 2.0000]$	$\lambda_{max} = 2.0000$
Собственные векторы:	$X_1 = [1.0000, 0.0000]^T$ $X_2 = [0.0000, 1.0000]^T$	$X_{max} = \begin{matrix}[0.0039] \\ [1.0000]\end{matrix}$
Количество итераций:	1	8

Тестовый пример 2

Вычислить с точностью 0.0001 собственные значения и собственные векторы матрицы **A** =

$$\begin{bmatrix} 1 & 0.007 \\ 0 & 2 \end{bmatrix}$$

Ответ:

Тип алгоритма	Вращений Якоби	Степенной
Собственные значения:	$\lambda = [1.0000, 2.0000]$	$\lambda_{max} = 2.0000$
Собственные векторы:	$X_1 = [1.0000, 0.0070]^T$ $X_2 = [-0.0070, 1.0000]^T$	$X_{max} = \begin{matrix}[0.0109] \\ [0.9999]\end{matrix}$
Количество итераций:	1	8

ЗАДАНИЕ
Вариант 7

Вычислить с точностью 0.0001 собственные значения и собственные векторы матрицы **A** =

$$\begin{bmatrix} 3.73 & 0.81 & 2.07 & 0.92 & -0.53 \\ 0.81 & 3.73 & 0.81 & 2.07 & 0.92 \\ 2.07 & 0.81 & 3.73 & 0.81 & 2.32 \\ 0.92 & 2.07 & 0.81 & 3.73 & -0.53 \\ -0.53 & 0.92 & 2.32 & -0.53 & 3.73 \end{bmatrix}$$

Ответ:

Метод вращений Якоби
Собственные значения: $\lambda = [0.1468, 1.6142, 3.8616, 5.1870, 7.8403]$
Собственные векторы: $X_1 = [0.4271, -0.2505, -0.6115, 0.2563, 0.5614]^T$ $X_2 = [0.3337, 0.6214, -0.3033, -0.6406, -0.0145]^T$ $X_3 = [0.6892, -0.5171, 0.2827, -0.2690, -0.3245]^T$ $X_4 = [0.2154, 0.2453, -0.3594, 0.5359, -0.6907]^T$ $X_5 = [0.4298, 0.4728, 0.5701, 0.4054, 0.3199]^T$
Количество итераций = 27

Степенной метод		
Собственное значение:	$\lambda_{max} =$	7.8402
Собственный вектор	$X_{max} =$	[0.4279] [0.4730] [0.5707] [0.4043] [0.3226]
Количество итераций	5	

Выводы

Таким образом, в ходе выполнения лабораторной работы были освоены метод вращений Якоби для вычисления собственных значений и собственных векторов вещественной симметричной матрицы, а также степенной метод поиска максимального по модулю собственного значения и соответствующего ему собственного вектора. Составлена компьютерная программа, на тестовых примерах проверена правильность её работы, с заданной точностью вычислены собственные значения и векторы матрицы заданного варианта.