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 253505
Снежко Максим Андреевич
Проверил: Анисимов Владимир Яковлевич

Минск 2023

Содержание

1. ЦЕЛЬ РАБОТЫ

1) Изучить метод вращений Якоби для симметрических матриц.
2) Составить алгоритм и программу нахождения собственных значений и собственных векторов методом Якоби.
3) Проверить правильность работы программы на тестовых примерах.
4) Решить задание заданного варианта.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Итерационный метод Якоби был предложен еще в середине 19-го века, однако долгое время не находил применения из-за слишком большого по тем временам объема вычислений. В настоящее время известно большое количество его модификаций, основная идея которых однако остается прежней. Из линейной алгебры известно, что всякая симметрическая матрица A может быть приведена к диагональному виду ортогональным преобразованием подобия

$$V^{-1} AV = \Lambda,$$

где Λ – диагональная матрица. При этом для ортогональной матрицы V справедливо условие $V^{-1}=V^{*}$, т.е. ортогональное преобразование подобия можно записать в виде

$$V^{*} AV = \Lambda. \tag{4.4}$$

Последнее условие дает фактически матричное уравнение, которое можно использовать для вычисления элементов матриц V и Λ. Однако метод Якоби 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

$$A^{(k)} \rightarrow A^{(k+1)} = V^{(k)*} A^{(k)} V^{(k)} = V^{(k)*} \dots V^{(0)*} A^{(0)} V^{(0)} \dots V^{(k)}, \qquad k=0,1,2\dots, \tag{4.5}$$

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

$$v_{ii}= v_{jj} = \cos\varphi \quad v_{ij} = - v_{ji} = -\sin\varphi \ , \tag{4.6}$$

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 $k \rightarrow \infty$.

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.

Отметим, что, если разложение (4.4) найдено, то легко указать правило нахождения собственных векторов. Действительно, если λ_i - i-й диагональный элемент матрицы Λ , тогда, как известно из линейной алгебры, координаты собственного вектора матрицы A соответствующего собственному значению λ_i совпадают с элементами i-го столбца матрицы V.

Теперь остается указать способ выбора матрицы $V^{(k)} = V^{(k)}_{ij}(\varphi)$ на k-м шаге и доказать сходимость метода.

Итак пусть есть матрица $A^{(k)}$. Найдем в ней максимальный по модулю недиагональный элемент $a_{ij}^{(k)}$. Поскольку матрица симметрическая, то можно считать, что $i<j$. Найдем значение угла поворота $\varphi = \varphi_k$ из условия равенства нулю элемента $a_{ij}^{(k+1)}$ матрицы

$$A^{(k+1)} = V^{(k)*} A^{(k)} V^{(k)}.$$

Положим $B = A^{(k)} V^{(k)}$. Тогда в виду определения матрицы поворота $V^{(k)} = V^{(k)}_{ij}\ (\varphi)$ элементы всех столбцов матрицы B, кроме i-го и j-го, совпадают с элементами матрицы $A^{(k)}$. Для элементов i-го и j-го столбцов имеем

$$\begin{aligned} b_{si} &= a_{si}^{(k)} \cos\varphi_k + a_{sj}^{(k)} \sin\varphi_k, \\ b_{sj} &= -a_{si}^{(k)} \sin\varphi_k + a_{sj}^{(k)} \cos\varphi_k, \quad s = 1, 2, ..., n. \end{aligned} \tag{4.7}$$

Аналогично матрица $A^{(k+1)} = V^{(k)*} B$ во всех строках, кроме i-ой и j-ой, имеет те же элементы, что и B. Элементы i-ой и j-ой строк имеют вид

$$\begin{aligned} a_{is}^{(k+1)} &= b_{is} \cos\varphi_k + b_{js} \sin\varphi_k, \\ a_{js}^{(k+1)} &= -b_{is} \sin\varphi_k + b_{js} \cos\varphi_k, \quad s = 1, 2, ..., n. \end{aligned} \tag{4.8}$$

Обратим внимание, что матрицы $A^{(k+1)}$ и $A^{(k)}$ различаются только суммой

$$[a_{is}^{(k+1)}]^2 + [a_{js}^{(k+1)}]^2 = b_{is}^2 + b_{js}^2 = [a_{is}^{(k)}]^2 + [a_{js}^{(k)}]^2$$

С учетом равенства $a_{ij}^{(k)} = a_{ji}^{(k)}$ из формул (4.7) и (4.8) получим

$$\begin{aligned} a_{ij}^{(k+1)} &= b_{ij} \cos\varphi_k + b_{jj} \sin\varphi_k = \\ &= (-a_{ii}^{(k)} \sin\varphi_k + a_{ij}^{(k)} \cos\varphi_k)\cos\varphi_k + (-a_{ji}^{(k)} \sin\varphi_k + a_{jj}^{(k)} \cos\varphi_k)\sin\varphi_k = \\ &= a_{ij}^{(k)} \cos 2\varphi_k + \frac{1}{2}(a_{jj}^{(k)} - a_{ii}^{(k)})\sin 2\varphi_k , \end{aligned} \tag{4.9}$$

Полагая в (4.9) $a_{ij}^{(k+1)} = 0$, получим

$$tg 2\varphi_k = 2a_{ij}^{(k)} / (a_{ii}^{(k)} - a_{jj}^{(k)}) \quad (-\pi/4 < \varphi_k < \pi/4)$$

или

$$\cos\varphi_k = \sqrt{\frac{1}{2}(1 + (1 + p_k^2))^{-1/2}}, \quad \sin\varphi_k = \operatorname{sgn} p_k \sqrt{\frac{1}{2}(1 - (1 + p_k^2))^{-1/2}}, \tag{4.10}$$

где

$$p_k = 2a_{ij}^{(k)} / (a_{ii}^{(k)} - a_{jj}^{(k)}).$$

Обозначим через $t(A)$ сумму квадратов всех недиагональных элементов матрицы A. Тогда

$$\begin{aligned} t(A^{(k+1)}) &= t(A^{(k)}) - 2[a_{ij}^{(k)}]^2 + \frac{1}{2}[(a_{jj}^{(k)} - a_{ii}^{(k)})\sin 2\varphi_k + 2a_{ij}^{(k)} \cos 2\varphi_k]^2 = \\ &= t(A^{(k)}) - 2[a_{ij}^{(k)}]^2 + \frac{1}{2}[2a_{ij}^{(k+1)}]^2 = t(A^{(k)}) - 2[a_{ij}^{(k)}]^2. \end{aligned} \tag{4.11}$$

Таким образом, значение функции $t(A)$ уменьшается на каждом шаге.

Покажем, что итерационный процесс в методе Якоби сходится. Действительно, в силу выбора элемента $a_{ij}^{(k)}$ справедлива оценка

$$t(A^{(k)}) \le n(n-1)[a_{ij}^{(k)}]^2,$$

откуда

$$[a_{ij}^{(k)}]^2 \ge t(A^{(k)})/n(n-1).$$

С учетом этого неравенства из формулы (4.11) получаем

$$t(A^{(k+1)}) = t(A^{(k)}) - 2[a_{ij}^{(k)}]^2 \le t(A^{(k)}) - \frac{2t(A^{(k)})}{n(n-1)} = qt(A^{(k)}),$$

где

$$q = 1 - \frac{2}{n(n-1)}.$$

Очевидно, что $0<q<1$ при порядке матрицы $n>2$. Таким образом, получаем

$$t(A^{(k)}) \le q^k t(A^{(0)}) \qquad k = 1, 2, \ldots .$$

Последнее означает, что

$$\lim_{k\to\infty} t(A^{(k)}) = 0$$

и, следовательно, итерационный процесс сходится.

В итоге получаем следующий алгоритм метода вращений:

1) в матрице $A^{(k)}$ $(k=0,1,2,\ldots)$ 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);

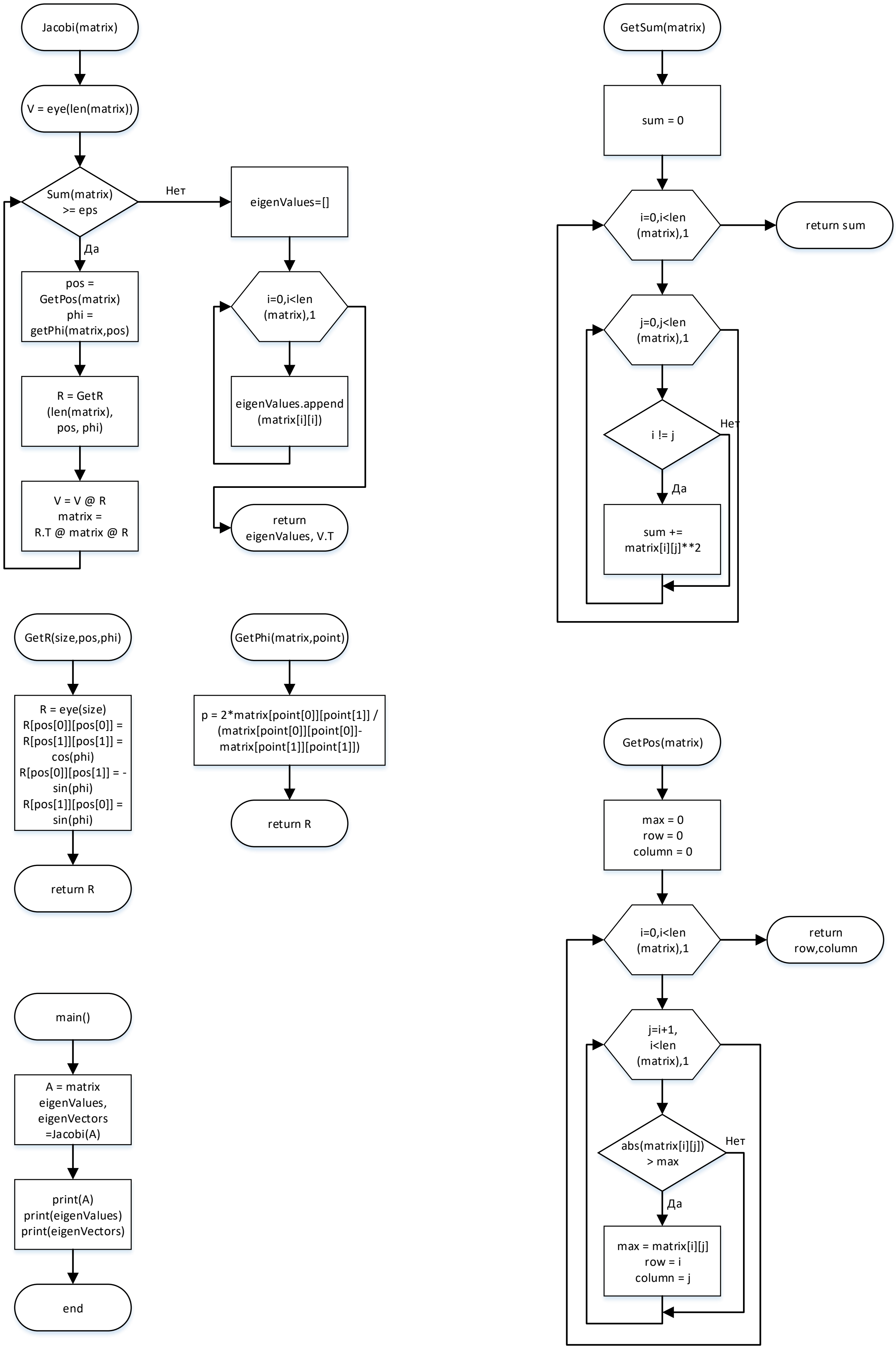
2) по формулам (4.10) вычисляем $\cos\varphi_k$ и $\sin\varphi_k$, далее используя формулы (4.7) и (4.8) находим элементы матрицы $A^{(k+1)}$;

3) итерационный процесс останавливаем, когда в пределах принятой точности величиной $t(A^{(k+1)})$ можно пренебречь;

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

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

3. *АЛГОРИТМ РЕШЕНИЯ*



4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

```
import numpy

eps = 0.0001

def CreateMatrix():
    k = 10
    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]
    ])
    return k * C + D

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

def GetPosOfMaxFromNodiag(matrix):
    max = 0.0
    row = 0
    column = 0
    for i in range(len(matrix)):
        for j in range(i + 1, len(matrix)):
            if abs(matrix[i][j]) > max:
                max = abs(matrix[i][j])
                row = i
                column = j

    return row, column
```

```
def GetPhi(matrix, point):
    if matrix[point[0]][point[0]] == matrix[point[1]][point[1]]:
        return numpy.pi / 4
    else:
        p = 2 * matrix[point[0]][point[1]] /
(matrix[point[0]][point[0]] - matrix[point[1]][point[1]])
    res = 0.5 * numpy.arctan(p)

    return res


def GetR(size, pos, phi):
    R = numpy.eye(size)
    R[pos[0]][pos[0]] = R[pos[1]][pos[1]] = numpy.cos(phi)
    R[pos[0]][pos[1]] = -numpy.sin(phi)
    R[pos[1]][pos[0]] = numpy.sin(phi)
    return R

def Jacobi(matrix):
    V = numpy.eye(len(matrix))
    while GetSumOfSqsFromNodiag(matrix) >= eps:

        posOfMax = GetPosOfMaxFromNodiag(matrix)
        phi = GetPhi(matrix, posOfMax)
        R = GetR(len(matrix), posOfMax, phi)
        V = V @ R
        matrix = R.T @ matrix @ R

    #print(matrix)

    eigenValues = []
    for i in range(len(matrix)):
        eigenValues.append(matrix[i][i])

    return numpy.array(eigenValues), V.T


def main():

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

    A = CreateMatrix()
    eigenValues, eigenVectors = Jacobi(A)

    print("Исходная матрица: \n", A)
    print("\nСобственные значения: \n", eigenValues)
    print("\nСобственные векторы: \n", eigenVectors)


main()
```

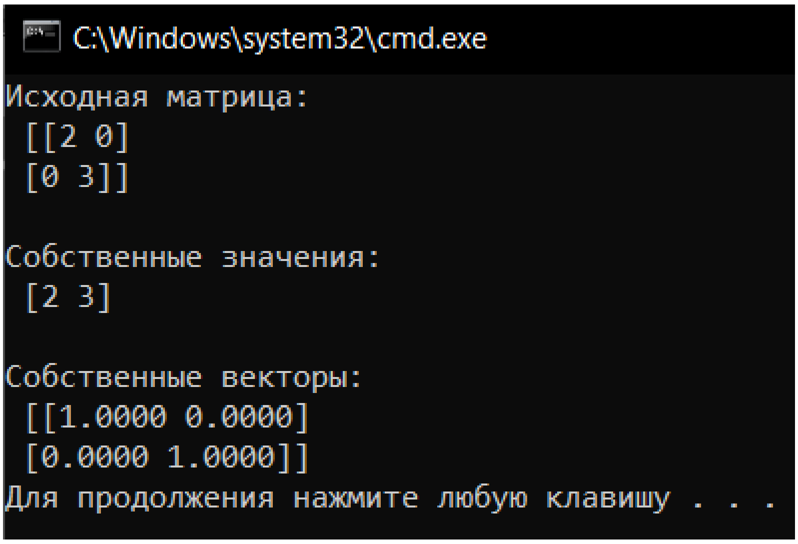
5. ТЕСТОВЫЕ ПРИМЕРЫ

Тестовый пример 1

Вычислить с точностью 0.0001 собственные значения и собственные векторы матрицы.

$$\boldsymbol{A} = [\mathbf{2\ 0\ 0\ 3}\,]$$

Результат работы программы:



Проверка в Maple:

```
A := matrix([
        [2, 0],
        [0, 3]
 ]);
```

$$A := \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \quad (1)$$

```
> eigen := evalf(eigenvectors(A), 5) :
>
> for i from 1 by 1 to 2 do eigen[i] end do;
```

$$[2., 1., \{[\,1.\ \ 0.\,]\}]$$
$$[3., 1., \{[\,0.\ \ 1.\,]\}] \quad (2)$$

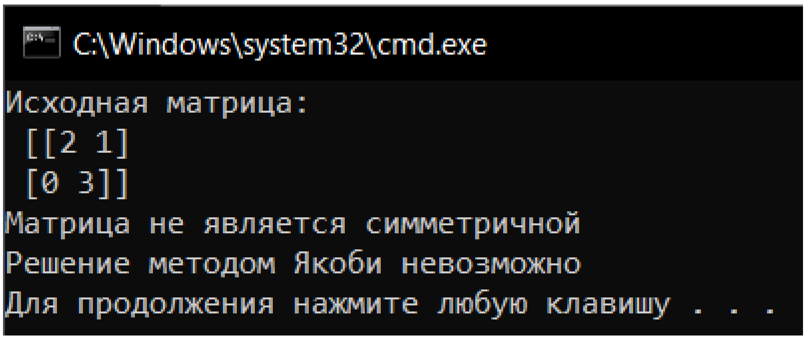
```
>
```

Тестовый пример 2

Вычислить с точностью 0.0001 собственные значения и собственные векторы матрицы.

$$\boldsymbol{A} = [\mathbf{2\ 1\ 0\ 3}\ ]$$

Результат работы программы:



Тестовый пример 3

Вычислить с точностью 0.0001 собственные значения и собственные векторы матрицы.

$$A = \mathbf{0.7\ 1\ 9\ 1\ 3\ 1\ 9\ 1\ 4}$$

Результат работы программы:

```
C:\Windows\system32\cmd.exe
Исходная матрица:
 [[0.7000 1.0000 9.0000]
 [1.0000 3.0000 1.0000]
 [9.0000 1.0000 4.0000]]

Собственные значения:
 [-6.8017  2.7744 11.7273]

Собственные векторы:
 [[ 0.7688 -0.0132 -0.6393]
 [-0.0918  0.9871 -0.1308]
 [ 0.6329  0.1593  0.7577]]
Для продолжения нажмите любую клавишу . 
```

Проверка в Maple:

```
A := matrix([
        [0.7, 1, 9],
        [1, 3, 1],
        [9, 1, 4]
 ]);
```

$$A := \begin{bmatrix} 0.7 & 1 & 9 \\ 1 & 3 & 1 \\ 9 & 1 & 4 \end{bmatrix} \qquad \textbf{(1)}$$

```
> eigen1 := evalf(eigenvectors(A), 5) :
>
> for i from 1 by 1 to 3 do eigen1[i] end do;
```

$$\left[11.728,\ 1.,\ \left\{\left[\ 0.63284\ \ 0.15930\ \ 0.75771\ \right]\right\}\right]$$

$$\left[-6.801,\ 1.,\ \left\{\left[\ -0.76880\ \ 0.013202\ \ 0.63936\ \right]\right\}\right]$$

$$\left[2.7741,\ 1.,\ \left\{\left[\ -0.091866\ \ 0.98712\ \ -0.13084\ \right]\right\}\right] \qquad \textbf{(2)}$$

Тестовый пример 4

Вычислить с точностью 0.0001 собственные значения и собственные векторы матрицы.

$$A = 0\,0.1\,2 \quad 3\,0.1\,0\,4 \quad 5\,2\,4\,0 \quad 6$$
$$3 \quad 5 \quad 6 \quad 0$$

Результат работы программы:

```
C:\Windows\system32\cmd.exe
Исходная матрица:
 [[0.0000 0.1000 2.0000 3.0000]
 [0.1000 0.0000 4.0000 5.0000]
 [2.0000 4.0000 0.0000 6.0000]
 [3.0000 5.0000 6.0000 0.0000]]

Собственные значения:
 [-0.0710 -4.4240 10.9176 -6.4226]

Собственные векторы:
 [[ 0.8786 -0.4749 -0.0439  0.0242]
 [ 0.3247  0.6605 -0.6751 -0.0508]
 [ 0.2748  0.4873  0.5631  0.6082]
 [-0.2171 -0.3174 -0.4745  0.7918]]
Для продолжения нажмите любую клавишу . .
```

Проверка в Maple:

```
A := matrix([
        [0, 0.1, 2, 3],
        [0.11, 0, 4, 5],
        [2, 4, 0, 6],
        [3, 5, 6, 0]
]);
```

$$A := \begin{bmatrix} 0 & 0.1 & 2 & 3 \\ 0.11 & 0 & 4 & 5 \\ 2 & 4 & 0 & 6 \\ 3 & 5 & 6 & 0 \end{bmatrix} \qquad (1)$$

```
> eigen1 := evalf(eigenvectors(A), 5) :
>
> for i from 1 by 1 to 4 do eigen1[i] end do;
```

$$[-4.4219, 1., \{[-0.31125 \;\; -0.63228 \;\; 0.64649 \;\; 0.04884]\}]$$
$$[-0.0749, 1., \{[-0.87688 \;\; 0.47292 \;\; 0.04466 \;\; -0.023572]\}]$$
$$[-6.4227, 1., \{[-0.18517 \;\; -0.27054 \;\; -0.40490 \;\; 0.67555]\}]$$
$$[10.916, 1., \{[-0.33700 \;\; -0.59816 \;\; -0.69081 \;\; -0.74607]\}] \qquad (2)$$

```
>
```

6. ЗАДАНИЕ

Вариант 10, k = 10

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:

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

```
C:\Windows\system32\cmd.exe
Исходная матрица:
 [[ 4.3300  0.8100  2.6700  0.9200 -0.5300]
 [ 0.8100  4.3300  0.8100  2.6700  0.9200]
 [ 2.6700  0.8100  4.3300  0.8100  2.9200]
 [ 0.9200  2.6700  0.8100  4.3300 -0.5300]
 [-0.5300  0.9200  2.9200 -0.5300  4.3300]]

Собственные значения:
 [ 4.6669  9.1894  6.2374  1.6199 -0.0636]

Собственные векторы:
 [[ 0.7280 -0.4219  0.1797 -0.1624 -0.4831]
 [ 0.4317  0.4483  0.5851  0.3877  0.3464]
 [-0.0657 -0.3961  0.3903 -0.5835  0.5882]
 [-0.2822 -0.6472  0.2587  0.6590  0.0147]
 [ 0.4469 -0.2127 -0.6372  0.2203  0.5481]]
Для продолжения нажмите любую клавишу . . .
```

> A := matadd(evalm(k·c), d);

$$A := \begin{bmatrix}4.33 & 0.81 & 2.67 & 0.92 & -0.53\\ 0.81 & 4.33 & 0.81 & 2.67 & 0.92\\ 2.67 & 0.81 & 4.33 & 0.81 & 2.92\\ 0.92 & 2.67 & 0.81 & 4.33 & -0.53\\ -0.53 & 0.92 & 2.92 & -0.53 & 4.33\end{bmatrix} \qquad (1)$$

> *eigen* := *evalf*(*eigenvectors*(*A*), 5) :

>

> **for** *i* **from** 1 **by** 1 **to** 5 **do** *eigen*[i] **end do**;

$[6.2373, 1., \{[-0.066035\ \ -0.39649\ \ 0.39064\ \ -0.58389\ \ 0.58734]\}]$

$[1.6194, 1., \{[-0.28215\ \ -0.64716\ \ 0.25886\ \ 0.65906\ \ 0.014504]\}]$

$[-0.0640, 1., \{[0.44663\ \ -0.21338\ \ -0.63707\ \ 0.21979\ \ 0.54847]\}]$

$[9.1883, 1., \{[-0.43153\ \ -0.44822\ \ -0.58499\ \ -0.38743\ \ -0.34712]\}]$

$[4.6668, 1., \{[-0.72824\ \ 0.42148\ \ -0.17983\ \ 0.16206\ \ 0.48326]\}] \qquad (2)$

7. ВЫВОД

Таким образом, в ходе выполнения лабораторной работы был изучен методы вращений Якоби для симметричных матриц, составлен алгоритм и программа нахождения собственных значений и собственных векторов методом Якоби, проверена правильность работы программы на тестовых примерах, численно решено задание заданного варианта.