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) |

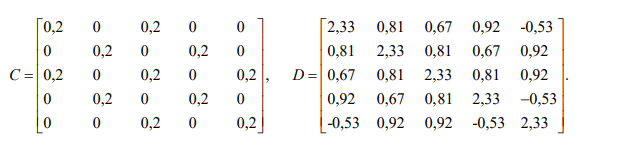
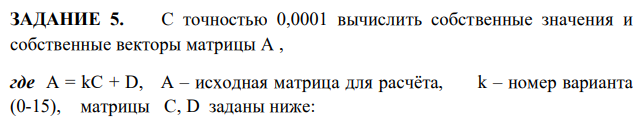
Минск 2024

**Содержание**

1. Цель работы
2. Задание
3. Программная реализация
4. Полученные результаты
5. Оценка полученных результатов
6. Вывод

**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

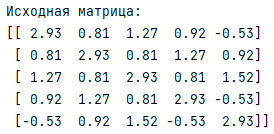


**Вариант 3**

**Программная реализация**

*Исходные данные*

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



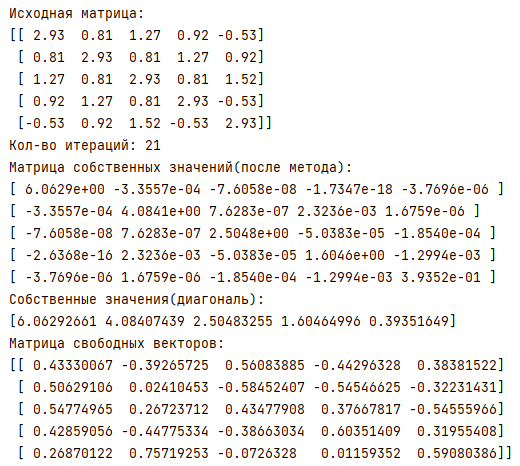
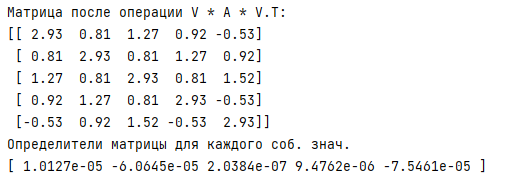
Код метода Якоби(вращений):

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

def check\_matrix\_simmetrix(matrix):  
 if(not numpy.all(matrix == matrix.transpose())):  
 print("Матрица не является симметричной!")  
 exit()

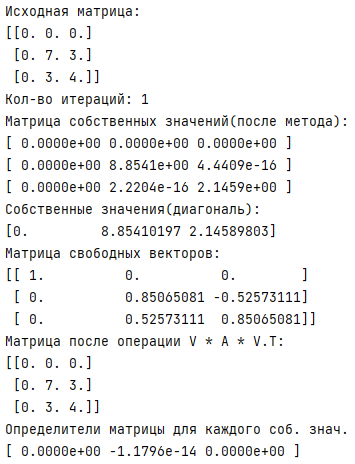
def not\_diag\_max\_element(matrix):  
 max = matrix[0][1]  
 max\_i = 0  
 max\_j = 1  
 for i in range(len(matrix)):  
 for j in range(len(matrix)):  
 if(j > i):  
 if(abs(matrix[i][j]) > abs(max)):  
 max = matrix[i][j]  
 max\_i = i  
 max\_j = j  
 return max\_i, max\_j  
  
  
def not\_diag\_squares(matrix):  
 sum = 0.0  
 for i in range(len(matrix)):  
 for j in range(len(matrix)):  
 if(j > i):  
 sum += 2 \* matrix[i][j] \*\* 2return sum

**Полученные результаты**

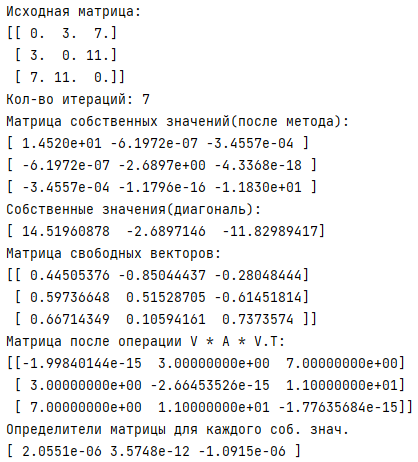
 

По значениям определителей матрицы для каждого собственного значения можно сделать вывод, что точность 0,0001 была достигнута.

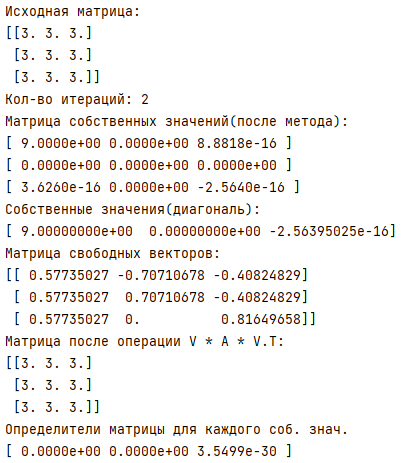
*Тестовый пример 1. Нулевые строка и столбец*

**

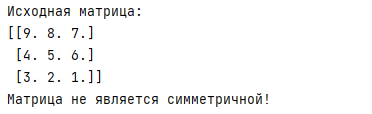
*Тестовый пример 2. Нулевая диагональ*



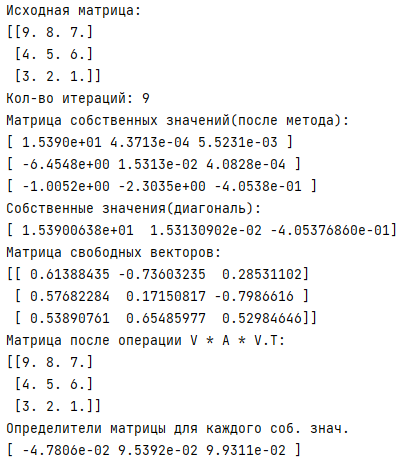
*Тестовый пример 3. Нулевая матрица*



*Тестовый пример 3. Несимметричная матрица*

**

*Тестовый пример 3. Несимметричная матрица(без проверки)*



**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.