МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования

«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

ОТЧЕТ по лабораторной работе № 6

по дисциплине

«Вычислительная математика»

Направление подготовки
02.03.01 «Математика и компьютерные науки»

Вариант № 6

Выполнил(а): студент гр. Б9122-02.03.01сцт
Бекболот Отгонцэцэг
Проверил: преподаватель
_____ Ф.И.О.

Владивосток
2024

Цель работы :

Найти методом простых итераций одно из собственных значений и все собственные значения симметричной положительно определенной матрицы A методом вращений (Якоби).

Ход работы:

1. Реализовать метод итераций по следующей схеме:
   a. Задаем нормированное начальное приближение x^0 и точность нахождения собственного значения ε;
   b. Находим значение вектора $y^{k+1} = Ax^k$;
   c. Осуществляем нормировку $x^{k+1} = y^{k+1} / \|y^{k+1}\|$.
   d. Вычисляем значение $\lambda^{k+1} = \max(x_i^{k+1} / x_i^k)$, где $i = 0, n - 1$, n – размерность матрицы A.
   e. Повторяем шаги $(a) - (d)$ до тех пор, пока не будет выполняться условие $|\lambda^{k+1} - \lambda^k| \leq \varepsilon$.
2. Реализовать метод вращений:
   a. Задаем точность нахождения собственных значений ε;
   b. Поскольку по условию исходная матрица симметричная, мы можем рассматривать либо верхний треугольник матрицы, либо нижний. Рассмотрим верхнюю треугольную наддиагональную часть матрицы A. В ней выделяем максимальный по модулю элемент $a_{ij}, i \neq j$;
   c. Найти угол поворота по формуле $\varphi = \frac{1}{2} * arctg(2a_{ij} / a_{ii} - a_{jj})$ (если $a_{ii} = a_{jj}$, то угол $\varphi = \frac{\pi}{4}$);
   d. Составить матрицу вращения T_{ij} , где:

$$t_{ii} = t_{jj} = \cos\varphi, t_{ij} = -t_{ji} = -\sin\varphi,$$

$$t_{kk} = 1, k \neq ii, ij, ji, jj$$

   Остальные элементы матрицы вращений равны нулю;

   e. Вычислить новое приближение $A^{k+1} = (T^k{}_{ij})^T A^k T^k{}_{ij}$

   Замечание: в целях оптимизации перемножать полностью на матрицы T не нужно, поскольку по факту у матрицы A поменяются только компоненты ii, ij, ji и jj;

   f. Повторять шаги $(a) - (e)$ до тех пор, пока не будет выполняться условие $\sum i \neq j |a^{k+1}{}_{ij}|^2 \leq \varepsilon$, то есть сумма квадратов внедиагональных элементов матрицы A^{k+1} не должна превышать заданную на первом шаге точность.

3. Необходимо сгенерировать симметричную, положительно определенную матрицу A /размерностей n = 3, n = 5, n = 7. Изучить, сколько требуется шагов обоим методам для каждого случая при ε = 10^{-3}, 10^{-7};
4. Вывести полученные собственные значения матрицы A^{k+1}.
5. Для каждой из матриц пункта 2 вывести число итераций, потребовавшихся для нахождения значений.

Теоретические сведения:

Метод вращений Якоби заключается в сведении исходной матрицы A к почти диагональной. Из курса линейной алгебры известно, что собственными значениями диагональной матрицы являются элементы на главной диагонали:

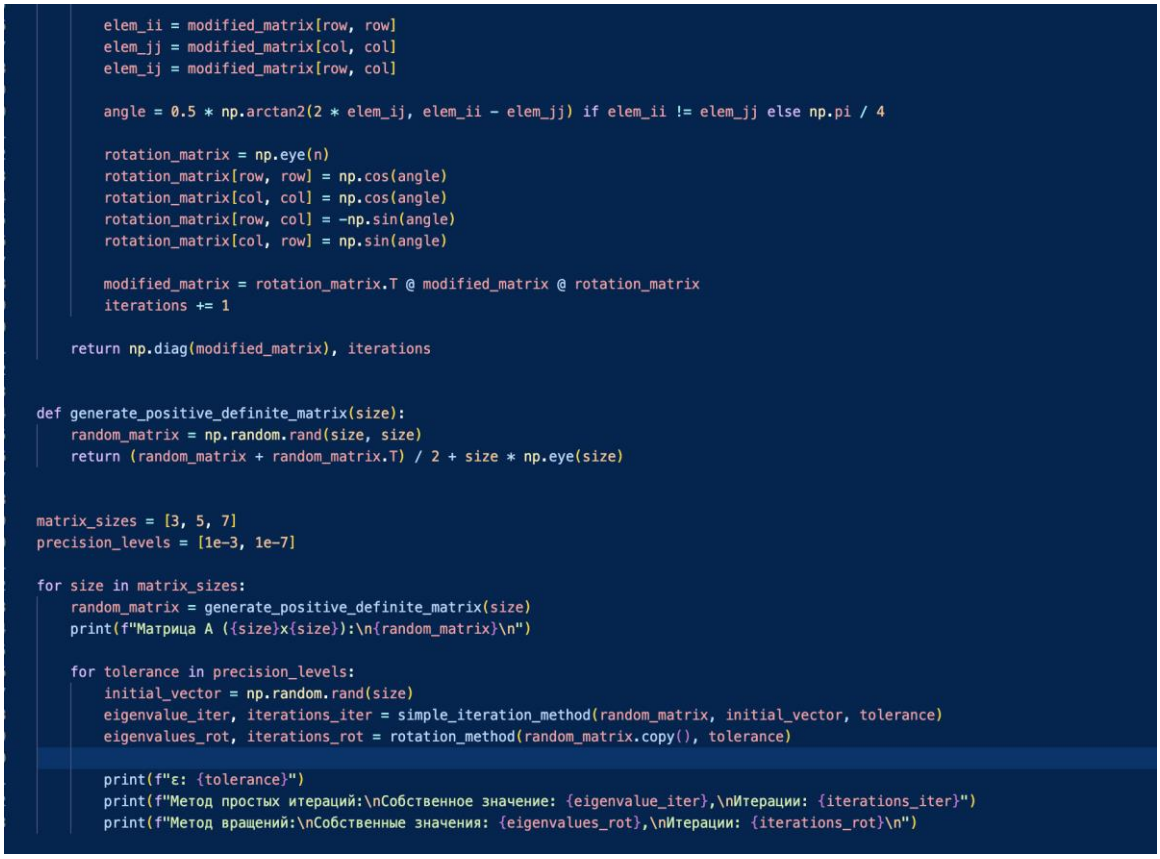
$$\lambda_i = a_{ii}, i = 1, \dots, n$$

Таким образом, если свести матрицу A к диагональному виду, то решением полной задачи собственных значений будет являться вектор с элементами:

$$\lambda = (a_{11}, a_{22}, \dots, a_{nn})$$

Код программы:

```python
import numpy as np

def simple_iteration_method(matrix, initial_vector, tolerance):
    n = matrix.shape[0]
    vector = initial_vector / np.linalg.norm(initial_vector)
    eigenvalue = 0
    iterations = 0

    while True:
        y = matrix @ vector
        vector_next = y / np.linalg.norm(y)


        eigenvalue_next = np.dot(vector_next, matrix @ vector_next) / np.dot(vector_next, vector_next)
        iterations += 1


        if np.abs(eigenvalue_next - eigenvalue) <= tolerance:
            break

        vector = vector_next
        eigenvalue = eigenvalue_next

    return eigenvalue, iterations


def rotation_method(matrix, tolerance):
    n = matrix.shape[0]
    modified_matrix = matrix.copy()
    iterations = 0

    while True:
        max_off_diagonal = 0
        row, col = -1, -1


        for r in range(n):
            for c in range(r + 1, n):
                if abs(modified_matrix[r, c]) > max_off_diagonal:
                    max_off_diagonal = abs(modified_matrix[r, c])
                    row, col = r, c

        if max_off_diagonal < tolerance:
            break

```



```
        elem_ii = modified_matrix[row, row]
        elem_jj = modified_matrix[col, col]
        elem_ij = modified_matrix[row, col]

        angle = 0.5 * np.arctan2(2 * elem_ij, elem_ii - elem_jj) if elem_ii != elem_jj else np.pi / 4

        rotation_matrix = np.eye(n)
        rotation_matrix[row, row] = np.cos(angle)
        rotation_matrix[col, col] = np.cos(angle)
        rotation_matrix[row, col] = -np.sin(angle)
        rotation_matrix[col, row] = np.sin(angle)

        modified_matrix = rotation_matrix.T @ modified_matrix @ rotation_matrix
        iterations += 1

    return np.diag(modified_matrix), iterations


def generate_positive_definite_matrix(size):
    random_matrix = np.random.rand(size, size)
    return (random_matrix + random_matrix.T) / 2 + size * np.eye(size)


matrix_sizes = [3, 5, 7]
precision_levels = [1e-3, 1e-7]

for size in matrix_sizes:
    random_matrix = generate_positive_definite_matrix(size)
    print(f"Матрица A ({size}x{size}):\n{random_matrix}\n")

    for tolerance in precision_levels:
        initial_vector = np.random.rand(size)
        eigenvalue_iter, iterations_iter = simple_iteration_method(random_matrix, initial_vector, tolerance)
        eigenvalues_rot, iterations_rot = rotation_method(random_matrix.copy(), tolerance)

        print(f"ε: {tolerance}")
        print(f"Метод простых итераций:\nСобственное значение: {eigenvalue_iter},\nИтерации: {iterations_iter}")
        print(f"Метод вращений:\nСобственные значения: {eigenvalues_rot},\nИтерации: {iterations_rot}\n")
```

Описание:

Программа представляет два метода для нахождения собственных значений симметрических положительно определённых матриц: метод простых итераций и метод вращений, а также предоставляет функции для генерации таких матриц.

Основные этапы работы программы:

1. Функция simple_iteration_method

- def simple_iteration_method(matrix, initial_vector, tolerance):

Эта функция реализует метод простых итераций для нахождения собственного значения симметричной положительно определенной матрицы.

Параметры:

- matrix: двумерный массив (матрица), для которой необходимо найти собственные значения.
- initial_vector: начальный вектор, использующийся для итерации.
- tolerance: уровень допустимой точности для остановки итераций.

Определяются размерность матрицы, нормализуется начальный вектор, и инициализируются переменные для собственного значения (eigenvalue) и счетчика итераций (iterations).
Затем запускается бесконечный цикл, который будет продолжаться до тех пор, пока не будет достигнута требуемая точность. Вектор *y* получается через умножение матрицы на текущий вектор. Затем vector_next) нормализуется для получения следующего вектора.

- eigenvalue_next = np.dot(vector_next, matrix @ vector_next) / np.dot(vector_next, vector_next)

Находится новое собственное значение, рассчитываемое через скалярное произведение.

- if np.abs(eigenvalue_next - eigenvalue) <= tolerance:break

Проверка на сходимость: если разница между новым и старым собственным значением меньше заданной точности, цикл прерывается.

- vector = vector_next ,eigenvalue = eigenvalue_next

Обновляются текущие векторы и собственные значения для следующей итерации.

- return eigenvalue, iterations

Возвращаются найденное собственное значение и общее количество итераций.

2. Функция rotation_method

- def rotation_method(matrix, tolerance):

Эта функция реализует метод вращений для нахождения собственных значений симметричной положительно определенной матрицы.

- max_off_diagonal = 0 row, col = -1, -1

Инициализация переменных для хранения максимального элемента вне диагонали и его индексов.

- for r in range(n): , for c in range(r + 1, n):

Цикл для поиска максимального элемента вне главной диагонали.

- if max_off_diagonal < tolerance: break

Если максимальное значение вне диагонали меньше допустимой точности, цикл прерывается.

- elem_ii = modified_matrix[row, row], elem_jj = modified_matrix[col, col],elem_ij = modified_matrix[row, col]

Получение значений элементов, необходимых для вычисления угла вращения.

- angle = 0.5 * np.arctan2(2 * elem_ij, elem_ii - elem_jj) if elem_ii != elem_jj else np.pi / 4

Вычисление угла вращения на основе значений элементов.

- rotation_matrix = np.eye(n) …

Создание матрицы вращения с использованием вычисленного угла.

- modified_matrix = rotation_matrix.T @ modified_matrix @ rotation_matrix, iterations += 1

Обновление матрицы через вращение и увеличение счетчика итераций.

- return np.diag(modified_matrix), iterations

Возврат собственных значений (в виде диагонали модифицированной матрицы) и количества итераций.

3. **Функция generate_positive_definite_matrix**

- def generate_positive_definite_matrix(size):

Эта функция генерирует симметричную положительно определенную матрицу заданного размера.

- random_matrix = np.random.rand(size, size)
- return (random_matrix + random_matrix.T) / 2 + size * np.eye(size)

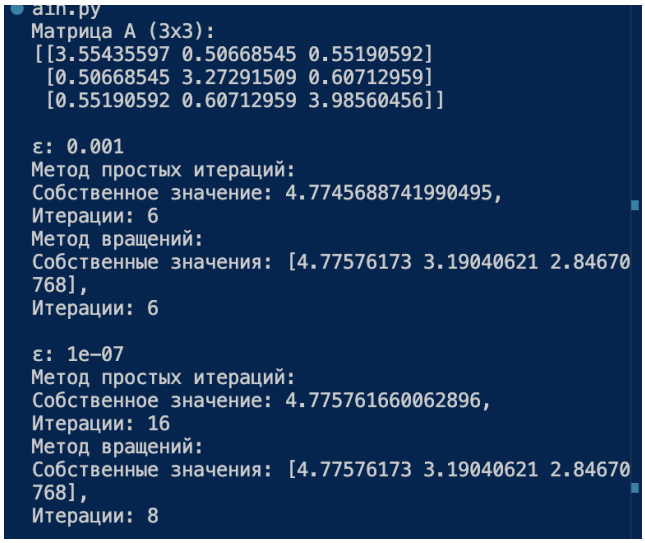
Создается случайная матрица, затем симметризуется путём усреднения с её транспонированной версией и добавляется диагональная матрица для обеспечения положительной определенности.

- В основной блоке программы:

Устанавливаются размеры матриц и уровни точности. Генерируются случайные симметричные положительно определённые матрицы.
Для каждой матрицы и каждого уровня точности применяются и выводятся результаты работы методов простых итераций и вращений.

Полученные результаты:

Для матрицы 3x3 с точностями 10-3 и 10-7:



Для матрицы 5x5 с точностями 10-3 и 10-7:

```
Матрица A (5x5):
[[5.37192878 0.8337251  0.60422669 0.48133401 0.6291
9392]
 [0.8337251  5.3479513  0.89724905 0.21447394 0.4096
2353]
 [0.60422669 0.89724905 5.50466861 0.55721452 0.2422
5865]
 [0.48133401 0.21447394 0.55721452 5.45096986 0.6039
0217]
 [0.62919392 0.40962353 0.24225865 0.60390217 5.7615
4751]]

ε: 0.001
Метод простых итераций:
Собственное значение: 7.691669316506557,
Итерации: 6
Метод вращений:
Собственные значения: [7.69322969 5.60113787 5.13107
298 4.67595547 4.33567005],
Итерации: 20

ε: 1e-07
Метод простых итераций:
Собственное значение: 7.693229619757281,
Итерации: 20
Метод вращений:
Собственные значения: [7.69322979 5.60113779 5.13107
299 4.67595544 4.33567005],
Итерации: 28
```

Для матрицы 7x7 с точностями 10-3 и 10-7:

```
Матрица A (7x7):
[[7.07448866 0.33007761 0.56365604 0.39921733 0.4416
1693 0.559742
  0.03958682]
 [0.33007761 7.96194199 0.63812217 0.60322457 0.4915
3961 0.29481087
  0.60731967]
 [0.56365604 0.63812217 7.40064703 0.73529828 0.3721
7444 0.11578628
  0.420149  ]
 [0.39921733 0.60322457 0.73529828 7.60848781 0.3152
2704 0.56424532
  0.32439092]
 [0.44161693 0.49153961 0.37217444 0.31522704 7.2042
7835 0.71072197
  0.29732651]
 [0.559742   0.29481087 0.11578628 0.56424532 0.7107
2197 7.0401206
  0.47096691]
 [0.03958682 0.60731967 0.420149   0.32439092 0.2973
2651 0.47096691
  7.39967321]]

ε: 0.001
Метод простых итераций:
Собственное значение: 10.131493819089673,
Итерации: 8
Метод вращений:
Собственные значения: [10.13302606  7.5241225   7.35
733275  7.08595245  6.90027948  6.58246475
  6.10645966],
Итерации: 39

ε: 1e-07
Метод простых итераций:
Собственное значение: 10.133026142325358,
Итерации: 26
Метод вращений:
Собственные значения: [10.13302632  7.52412258  7.35
733327  7.08595247  6.90027935  6.5824646
  6.10645907],
Итерации: 56
```

Выводы:

В программе два метода для нахождения собственных значений симметричных положительно определённых матриц: метод простых итераций и метод вращений.

1. **Метод простых итераций:**
   Этот метод использует итеративный подход, начиная с произвольно выбранного вектора, и в каждой итерации находит новый вектор, который нормализуется, что позволяет постепенно приближаться к собственному значению самой матрицы.
   Метод демонстрирует хорошую сходимость для симметричных положительно определённых матриц, однако его эффективность может варьироваться в зависимости от выбора начального вектора.
2. **Метод вращений:**
   Этот метод ориентирован на прямое преобразование матрицы через последовательные вращения, чтобы убрать все элементы вне главной диагонали, тем самым максимизируя элементы на диагонали.
   Метод не требует начальных предположений и подходит для получения всех собственных значений матрицы.

3. **Результаты эксперимента:**
   В ходе эксперимента были проведены вычисления для симметричных положительно определённых матриц размером 3x3, 5x5 и 7x7. Для каждого размера были установлены два уровня точности: (1e-3, 1e-7)
   Метод простых итераций – Количество итераций, необходимых для достижения заданной точности, варьировалось в зависимости от сложности матрицы и начального вектора.В среднем, метод показывал относительно быстрое сходимость, особенно для маленьких размеров матриц, но в случае больших матриц количество итераций могло значительно увеличиваться.
   Метод вращений - Метод вращений позволял вычислить все собственные значения одновременно, что является его значительным преимуществом. Он оставался стабильным и эффективным в расчётах независимо от начальных условий. Однако, как правило, метод вращений требовал больше вычислительных ресурсов, чем метод простых итераций, особенно для больших матриц из-за необходимости многократных преобразований.

4. **Сравнение методов:**
   - Сходимость:

Оба метода обеспечивают хорошую сходимость для симметричных положительно определённых матриц, но метод простых итераций показывает различные результаты в зависимости от начального вектора.

Метод вращений демонстрирует стабильную сходимость без зависимости от первоначальных условий.

- Количество собственных значений:

Метод простых итераций позволяет находить только одно собственное значение за итерацию, в то время как метод вращений находит все собственные значения одновременно.

- Эффективность:

Метод простых итераций может быть быстрее для малых матриц и при удачном выборе начального вектора. Однако, для больших и сложных матриц его эффективность может значительно снизиться.

Метод вращений требует больше вычислительных ресурсов, но компенсируется его универсальностью и способностью быстро находить все собственные значения.

5. Заключение:

В результате проведённого эксперимента было показано, что оба метода эффективно решают задачу нахождения собственных значений симметричных положительно определённых матриц. Каждый из методов имеет свои преимущества и недостатки, и выбор между ними зависит от конкретной задачи.