МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Дальневосточный федеральный университет»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
Департамент математического и компьютерного моделирования

ОТЧЕТ

по лабораторной работе

по дисциплине «Вычислительная математика»
по направлению подготовки «01.03.02 Прикладная математика и информатика»
шифр и название направления

образовательная программа «Математика и компьютерные технологии»
название образовательной программы

на тему **«Метод простой итерации»**

Выполнил студент гр. Б9122
Пелагеев Даниил Иванович

Проверил
Журавлев Павел Викторович

_____
(оценка)

г. Владивосток
2024

Оглавление

Введение

В данном отчёте рассматривается применение метода простой итерации для приближённого вычисления наибольшего по модулю собственного значения и соответствующего собственного вектора квадратной матрицы A. Целью работы является практическая реализация данного метода и анализ его сходимости, а также оценка качества приближений на тестовых примерах.

Постановка задачи

Дана квадратная матрица $A \in \mathbb{R}^{n \times n}$. Необходимо найти собственное значение λ наибольшего по модулю и соответствующий ему собственный вектор $\vec{x}$, т.е. решить следующую задачу:

$$A\vec{x} = \lambda\vec{x}.$$

Теоретическое описание метода

В основе метода простой итераци[1] для нахождения наибольшего по модулю собственного значения лежит итерационный процесс:

$$\lambda\vec{x} = A\vec{x}.$$

Для удобства введём вспомогательный вектор:

$$\vec{y} = A\vec{x}.$$

Тогда уравнение для собственного значения можно переписать в виде:

$$\lambda\vec{x} = \vec{y}.$$

Обозначим начальное приближение собственного вектора через $\vec{x}^{(0)}$, нормированное так, что $|\vec{x}^{(0)}| = 1$. На первой итерации вычисляем:

$$\vec{y}^{(1)} = A\vec{x}^{(0)}.$$

Далее для приближения собственного значения $\lambda^{(1)}$ умножаем обе части уравнения $\lambda\vec{x} = \vec{y}$ скалярно на $\vec{x}^{(0)}$:

$$\lambda = \frac{\vec{y} \cdot \vec{x}^{(0)}}{\vec{x}^{(0)} \cdot \vec{x}^{(0)}} = \vec{y}^{(1)} \cdot \vec{x}^{(0)},$$

учитывая, что $\|\vec{x}^{(0)}\| = 1$.

Следующее приближение собственного вектора вычисляем нормированием $\vec{y}^{(1)}$:

$$\vec{x}^{(1)} = \frac{\vec{y}^{(1)}}{\|\vec{y}^{(1)}\|}.$$

В общем случае итерационный процесс выглядит так:

$$\vec{y}^{(k+1)} = A\vec{x}^{(k)}, \quad k = 0, 1, \ldots$$

$$\lambda^{(k+1)} = \vec{y}^{(k+1)} \cdot \vec{x}^{(k)}.$$

$$\vec{x}^{(k+1)} = \frac{\vec{y}^{(k+1)}}{\|\vec{y}^{(k+1)}\|}.$$

Итерации продолжаются до тех пор, пока изменения в приближениях собственного значения и собственного вектора не станут несущественными. В качестве критерия остановки можно использовать близость направлений векторов:

$$\text{sign}(\lambda^{(k+1)})\vec{x}^{(k+1)} \approx \vec{x}^{(k)}.$$

Приближённо найденное по окончании итерационного процесса значение λ будет наибольшим по модулю собственным значением матрицы A, а $\vec{x}$ — соответствующим собственным вектором.

Ограничения на входные данные для быстрой сходимости метода состоят в том, что начальный вектор $\vec{x}^{(0)}$ не должен быть ортогонален к искомому собственному вектору. Если начальный вектор "близок"к искомому собственному вектору, итерационный процесс сходится быстро.

Практическая часть

Описание реализации

Был написан программный код на Python, который реализует описанный выше итерационный процесс. Также для проверки было использовано нахождение собственных значений с помощью библиотеки NumPy. В качестве начального приближения используется единичный вектор размерности n. Затем этот вектор преобразуется по формуле: $\frac{\mathbf{x}}{\|\mathbf{x}\|_2}$. Для матрицы A и начального приближения $\vec{x}^{(0)}$ алгоритм на каждом шаге выполняет следующие действия:

а) Вычисляет $\vec{y}^{(k+1)} = A\vec{x}^{(k)}$;
б) Находит $\lambda^{(k+1)} = \vec{y}^{(k+1)} \cdot \vec{x}^{(k)}$;
в) Нормирует $\vec{x}^{(k+1)} = \frac{\vec{y}^{(k+1)}}{\|\vec{y}^{(k+1)}\|}$;
г) Проверяет критерий остановки.

Тестовые примеры

Для тестирования метода были использованы следующие матрицы:

Пример 1.

$$A = \begin{pmatrix} -0.168700 & 0.353699 & 0.008540 & 0.733624 \\ 0.353699 & 0.056519 & -0.723182 & -0.076440 \\ 0.008540 & -0.723182 & 0.015938 & 0.342333 \\ 0.733624 & -0.076440 & 0.342333 & -0.045744 \end{pmatrix}, \epsilon = 1e-5.$$

Результат 1.

Матрица №1

A:

——— ——— ——— ———

-0.1687 0.353699 0.00854 0.733624

0.353699 0.056519 -0.723182 -0.07644

0.00854 -0.723182 0.015938 0.342333

0.733624 -0.07644 0.342333 -0.045744

——— ——— ——— ———

Метод завершился за 158 итераций.

Собственные значения матрицы: [-0.94356786 -0.74403641 0.68784308 0.85777418]

Наибольшее собственное значение по модулю: -0.9435678554868709

Вектор x: [-0.72360714 0.33173539 0.04102097 0.60387219]

Проверка вида $Ax - \lambda * x$: 8.211011126773054e-07

Пример 2.

$$A = \begin{pmatrix} 1.00 & 0.42 & 0.54 & 0.66 \\ 0.42 & 1.00 & 0.32 & 0.44 \\ 0.54 & 0.32 & 1.00 & 0.22 \\ 0.66 & 0.44 & 0.22 & 1.00 \end{pmatrix}, \epsilon = 1e-5.$$

Результат 2.

Матрица №2

A:

— — — —

1 0.42 0.54 0.66

0.42 1 0.32 0.44

0.54 0.32 1 0.22

0.66 0.44 0.22 1

— — — —

Метод завершился за 11 итераций.

Собственные значения матрицы: [2.3227488 0.24226071 0.6382838 0.79670669]

Наибольшее собственное значение по модулю: 2.3227488000697916

Вектор x: [0.5796425 0.45999662 0.43345942 0.5143254]

Проверка вида $Ax - \lambda * x$: 5.778797665781721e-07

Пример 3.

$$A = \begin{pmatrix} 2.2 & 1 & 0.5 & 2 \\ 1 & 1.3 & 2 & 1 \\ 0.5 & 2 & 0.5 & 1.6 \\ 2 & 1 & 1.6 & 2 \end{pmatrix}, \epsilon = 1e - 5.$$

Результат 3.

Матрица №3

A:

— — — —

2.2 1 0.5 2

1 1.3 2 1

0.5 2 0.5 1.6

2 1 1.6 2

— — — —

Метод завершился за 10 итераций.

Собственные значения матрицы: [5.65203233 1.54541834 0.22263593 -1.42008659]

Наибольшее собственное значение по модулю: 5.652032331760144

Вектор x: [0.53173592 0.44619424 0.40881571 0.59248403]

Проверка вида $Ax - \lambda * x$: 1.1967546839871334e-06

Заключение

В результате работы был реализован и исследован метод простой итерации для нахождения наибольшего по модулю собственного значения и соответствующего собственного вектора матрицы A. На практике метод показал быструю сходимость при удачном выборе начального приближения. При неблагоприятном выборе начального вектора сходимость может быть замедлена или потребовать большего числа итераций.

Список использованных источников

1. *Колобов, Александр Георгиевич.* Численные методы линейной алгебры: Методические указания и задания для студентов математических специальностей / Александр Георгиевич Колобов, Лилия Александровна Молчанова. — Издательство Дальневосточного университета, 2008. — Дата обращения: 10.12.2024.

Приложение А Пример кода

Ниже приведён пример кода на языке Python:

```
import numpy as np
from tabulate import tabulate


def simple_iteration(A, tol=1e-5, max_iter=1000):
    x = np.ones(A.shape[0])
    x = x / np.linalg.norm(x, ord=2)
    eigenvalue_old = 0
    iter_count = 0

    for _ in range(max_iter):
        iter_count += 1

        y = A @ x
        eigenvalue = y @ x
        x_new = y / np.linalg.norm(y)

        if (np.linalg.norm(np.sign(eigenvalue) * x - x_new) < tol
                and abs(eigenvalue - eigenvalue_old) < tol):
            print(f"The method completed in {iter_count} iterations.")
            return eigenvalue, x_new

        x = x_new
        eigenvalue_old = eigenvalue

    print(f"The maximum iterations ({max_iter}) without the desired
       accuracy is reached.")
    return eigenvalue_old, x


def main():
    tol = 1e-6
    matrices = [
        np.array(
            [
                [-0.168700, 0.353699, 0.008540, 0.733624],
                [0.353699, 0.056519, -0.723182, -0.076440],
                [0.008540, -0.723182, 0.015938, 0.342333],
                [0.733624, -0.076440, 0.342333, -0.045744]
            ]
        ),
        np.array(
            [
```

```
                [1.00, 0.42, 0.54, 0.66],
                [0.42, 1.00, 0.32, 0.44],
                [0.54, 0.32, 1.00, 0.22],
                [0.66, 0.44, 0.22, 1.00]
            ]
        ),
        np.array(
            [
                [2.2, 1, 0.5, 2],
                [1, 1.3, 2, 1],
                [0.5, 2, 0.5, 1.6],
                [2, 1, 1.6, 2]
            ]
        )
    ]

    for i in range(len(matrices)):
        print(f"Matrix num {i+1}")
        print("A:", tabulate(matrices[i]), sep="\n")
        eigenvalue, x = simple_iteration(matrices[i], tol)
        print("Eigenvalues of the matrix:", np.linalg.eigvals(matrices[i]))
        print("The largest eigenvalue modulo:", eigenvalue)
        print("Vector x:", x)
        # Ax - lmd * x
        print("Validation:", np.linalg.norm(matrices[i] @ x - eigenvalue *
            x), end="\n\n")


if __name__ == '__main__':
    main()
```