МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

ЛАБОРАТОРНАЯ РАБОТА №10
По основной образовательной программе подготовки бакалавров
направлению 01.03.02 Прикладная математика и информатика
профиль «Системное программирование»

Студент группы Б9122-01.03.02сп(4)
Кириенко Денис Олегович
_____
(подпись)
«_____» _____ 2024 г.

Преподаватель _старший преподаватель__
(должность, ученое звание)
Журавлев Павел Викторович___
_____
(подпись) (ФИО)
«_____» _____2024 г.

г. Владивосток
2024

Постановка задачи

Реализовать и протестировать метод “вращения с преградами” для решения полной проблемы собственных значений.

Теоретическое описание метода

Метод предназначен для решения полной проблемы собственных значений невырожденной симметрической матрицы A. Решается она с помощью сходящихся итерационных процессов. Входные данные:

1. Невырожденная симметрическая матрица A;
2. Положительное число p, определяющее точность решения.

Результатом работы метода является диагональная матрица D, на диагонали которой расположены все собственные значения данной матрицы.

Практическая часть

На каждой итерации строится матрица вращения T_{ij}, посредством которой происходит переход в следующую итерацию: $A^{m+1}=T_{ij}^{T}A^{m}T_{ij.}$

В методе осуществляется два цикла: общий — с итератором k, и вложенный — с итератором m. Для построения матрицы вращения T_{ij} необходимо выполнение нескольких шагов:

1. Вычислить преграду σ_k

Вычисление преграды имеет несколько реализаций. В данной работе преграда на шаге k вычислялась так: $\sigma_k = \sqrt{(\max| a_{ij}^{(m)}|)} * 10^{-k}$, где k — шаг, m — номер итерации матрицы A, а $a_{ij}^{(m)}$ — всякое значение матрицы $A^{(m)}$.

2. Найти значения i и j

Парой индексов (i, j) обозначается наибольший по модулю недиагональный элемент матрицы $A^{(m)}$, который соответствует условию $|a_{ij}^{(m)}| \geqslant \sigma_k$.

3. Найти значения c и s

Значения c и s должны быть такими, что $s^2 + c^2 = 1$. Вычисляются они по заданным формулам:

$$c = \sqrt{\frac{1}{2}\left(1 + \frac{|a_{ii}^{(m)} - a_{jj}^{(m)}|}{d}\right)},$$

$$s = \operatorname{sgn}[a_{ij}^{(m)}(a_{ii}^{(m)} - a_{jj}^{(m)})]\sqrt{\frac{1}{2}\left(1 - \frac{|a_{ii}^{(m)} - a_{jj}^{(m)}|}{d}\right)},$$

$$d = \sqrt{(a_{ii}^{(m)} - a_{jj}^{(m)})^2 + 4(a_{ij}^{(m)})^2}.$$

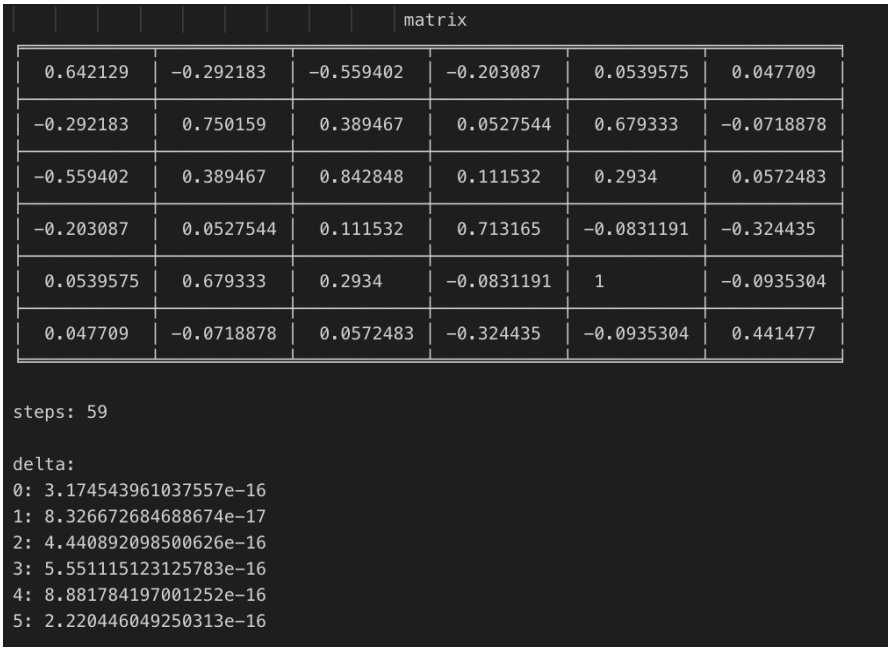
* (i, j) — пара индексов, найденная на шаге 2.

После выполнения указанных шагов строится матрица T_{ij}. Она основывается на единичной матрице, однако отличие матрицы вращения от единичной состоит в том, что:

1. Значение на позиции (i, i) равняется c;
2. Значение на позиции (i, j) равняется -s;
3. Значение на позиции (j, i) равняется s;
4. Значение на позиции (j, j) равняется c.

После этого находится $A^{m+1}=T_{ij}^{T}A^{m}T_{ij}$ и так происходит переход на следующую итерацию. Итерации заканчиваются на шаге m, если не удается найти наибольший по модулю недиагональный элемент матрицы $A^{(m)}$, модуль которого больше либо равен текущей прегаде, то есть элемент с шага 2 построения матрицы вращения. По окончании всех итераций должна получится диагональная матрица $D = A^{(m)}$, состоящая из собственных значений исходной матрицы A.

Работа выполнялась посредством языка программирования python и математической библиотеки numpy. Тестирование проводилось с параметром точности p равным 8 при различных размерах исходной матрицы. Результаты можно увидеть на фотографиях ниже. Количество итераций, обозначенное именем steps, в среднем на матрице 6x6 равняется 50-70, а на матрице 20x20 в среднем 750-850 при сохранении средней точности в e-15.



matrix

0.642129	-0.292183	-0.559402	-0.203087	0.0539575	0.047709
-0.292183	0.750159	0.389467	0.0527544	0.679333	-0.0718878
-0.559402	0.389467	0.842848	0.111532	0.2934	0.0572483
-0.203087	0.0527544	0.111532	0.713165	-0.0831191	-0.324435
0.0539575	0.679333	0.2934	-0.0831191	1	-0.0935304
0.047709	-0.0718878	0.0572483	-0.324435	-0.0935304	0.441477

steps: 59

delta:
0: 3.174543961037557e-16
1: 8.326672684688674e-17
2: 4.440892098500626e-16
3: 5.551115123125783e-16
4: 8.881784197001252e-16
5: 2.220446049250313e-16

matrix

0.64973	-0.320485	-0.0463471	-0.29205	-0.0399705	0.14987	0.203826	0.23909	0.0957142	0.0304222	0.153419	0.255461	-0.0513547	-0.0043939	0.0843
-0.320485	1	0.081136	0.069101	-0.37925	-0.26589	-0.364591	-0.111312	-0.0810856	0.0140493	0.0843136	0.0977557	-0.270653	0.248922	0.1877
-0.0463471	0.081136	0.719763	-0.0637672	-0.12139	0.146924	-0.0943761	0.0643247	0.0642429	-0.184103	0.0611356	0.174037	0.101121	0.0323847	0.0915
-0.29205	0.069101	-0.0637672	0.743348	-0.151804	0.158673	-0.266702	0.0718641	-0.0575254	-0.155161	-0.110016	-0.0926568	0.129182	-0.268657	-0.2055
-0.0399705	-0.37925	-0.12139	-0.151804	0.794838	-0.0612033	0.209833	-0.187743	-0.201787	0.0585522	0.015722	-0.316729	0.137234	0.101814	-0.3867
0.14987	-0.26589	0.146924	0.158673	-0.0612033	0.672732	0.0302367	0.0562833	-0.134959	-0.113208	0.263156	0.139031	0.177213	-0.0504945	-0.0260
0.203826	-0.364591	-0.0943761	-0.266702	0.209833	0.0302367	0.953348	-0.00738837	-0.192263	0.237858	0.0278112	0.216892	-0.0396018	0.137631	0.0387
0.23909	-0.111312	0.0643247	0.0718641	-0.187743	0.0562833	-0.00738837	0.603712	0.154197	0.229291	0.0168288	0.21389	-0.178752	-0.335046	0.2253
0.0957142	-0.0810856	0.0642429	-0.0575254	-0.201787	-0.134959	-0.192263	0.154197	0.622245	0.0533786	-0.0535606	0.117881	-0.143801	-0.339523	0.1377
0.0304222	0.0140493	-0.184103	-0.155161	0.0585522	-0.113208	0.237858	0.229291	0.0533786	0.835931	-0.292584	-0.0213249	-0.554333	0.000469898	0.1874
0.153419	0.0843136	0.0611356	-0.110016	0.015722	0.263156	0.0278112	0.0168288	-0.0535606	-0.292584	0.894685	0.381457	0.00171795	-0.101542	0.0594
0.255461	0.0977557	0.174037	-0.0926568	-0.316729	0.139031	0.216892	0.21389	0.117881	-0.0213249	0.381457	0.767088	-0.162991	-0.0458726	0.1789
-0.0513547	-0.270653	0.101121	0.129182	0.137234	0.177213	-0.0396018	-0.178752	-0.143801	-0.554333	0.00171795	-0.162991	0.698437	-0.0240038	-0.2530
-0.0043939	0.248922	0.0323847	-0.268657	0.101814	-0.0504945	0.137631	-0.335046	-0.339523	0.000469898	-0.101542	-0.0458726	-0.0240038	0.672072	0.0484
0.0843615	0.187727	0.0915711	-0.205563	-0.386799	-0.0260761	0.0387838	0.225321	0.137787	0.187468	0.0594154	0.178956	-0.253027	0.0484417	0.7722
0.0251176	0.272625	0.299981	-0.25778	-0.118551	-0.0794102	0.0540784	-0.173948	-0.100237	-0.134275	0.0165309	0.21529	0.0976608	0.252761	-0.1239
0.138525	-0.282547	0.0500729	0.0343261	0.0450967	0.0342217	0.0377696	0.17965	0.165279	-0.230302	0.125219	0.046565	0.18258	-0.419937	-0.1135
0.0584597	-0.343032	0.115582	0.0377753	0.23592	0.0678123	-0.0390027	0.0217317	0.110214	-0.17344	-0.167357	-0.141722	0.433297	-0.000680865	-0.0416
0.0822571	-0.1847	0.113408	0.0121453	-0.0779627	0.197812	-0.319871	0.00745372	0.160537	-0.245537	-0.00452206	-0.0222948	0.258548	-0.0481518	0.0743
-0.0952849	0.108844	0.294133	-0.322235	0.0490725	0.0108756	-0.12704	-0.22242	-0.00736152	-0.0722219	0.025884	-0.0727567	0.107796	0.268109	-0.0493

```
steps: 833

delta:
0: 2.287829214270287e-15
1: 7.698702786385078-15
2: 6.2450045135165055e-15
3: 1.5265566588595902e-15
4: 1.5265566588595902e-15
5: 1.1379786002407855e-15
6: 0.0
7: 8.326672684688674e-16
8: 4.996003610813204e-16
9: 2.3869795029440866e-15
10: 1.8873791418627666-15
11: 1.3322676295501878e-15
12: 5.551115123125783e-16
13: 1.1102230246251565e-16
14: 1.4432899320127035e-15
15: 1.5543122344752192e-15
16: 1.7763568394002505e-15
17: 1.021405182655144e-14
18: 4.884981308350689e-15
19: 1.7763568394002505e-15
```

Уточнения:

- Параметр delta обозначает разницу вычисленного значения со значением, полученным из numpy;
- Под средней точностью подразумевается среднее значение всех порядков различия вычисленных значений со значениями, данными библиотекой numpy.

Заключение

Метод был реализован и протестирован. Результаты тестирования кода метода дают неплохую среднюю точность e-16.

Приложение

```python
import sys

import numpy as np

from labs.funcs import *

sys.stdout = open("./labs/output.txt", "w")
```

```
def rotation_with_barriers(
    A: np.ndarray,
    p: int = 4,
) -> np.ndarray:
    D = A.copy()
    n = D.shape[0]

    if np.linalg.det(D) == 0:
        raise ValueError("matrix is singular")

    counter = 0

    for K in range(1, p + 1):
        sigma = np.sqrt(np.max(np.abs(np.diag(np.diag(M))))) * 10 ** (-K)
        # sigma = 10 ** (-K)

        while True:
            if counter > 1e5:
                raise ValueError("inf cycle")

            mx_val = -np.inf
            idx = ()

            for i in range(n):
                for j in range(n):
                    if D[i, j] > mx_val and np.abs(D[i, j]) >= sigma and i != j:
                        mx_val = D[i, j]
                        idx = (i, j)

            if mx_val == -np.inf:
                break

            i, j = idx[0], idx[1]

            d = np.sqrt((D[i, i] - D[j, j]) ** 2 + 4 * D[i, j] ** 2)
            s = np.sign(D[i, j] * (D[i, i] - D[j, j])) * np.sqrt(
                1 / 2 * (1 - np.linalg.norm(D[i, i] - D[j, j]) / d)
            )
            c = np.sqrt(1 / 2 * (1 + np.linalg.norm(D[i, i] - D[j, j]) / d))

            # print(f"K: {K} \nsigma: {sigma} \ni,j: {i+1,j+1} \nmx_val: {mx_val}")
            # print(f"c:\t{c}\ts:\t{s}")
            # print_matrix(D)

            T = np.eye(n)
            T[i, i] = T[j, j] = c
            T[i, j] = -s
            T[j, i] = s
```

```
            D = T.T @ D @ T

            counter += 1

    print(f"steps: {counter}")

    return np.diag(D)


size = (20, 20)
M = generate_symmetric_matrix(*size).astype(np.double)
M /= np.max(M)

# print_matrix(M, "matrix")

ans = rotation_with_barriers(M, p=8)  # max(p)=8
np_ans = np.linalg.eigvals(M)

ans = np.array(sorted(ans))
np_ans = np.array(sorted(np_ans))

print(
    f"""
delta:
{''.join(f"{i[0]}: {abs(i[1] - ans[i[0]])}\n" for i in enumerate(np_ans))}
"""
)
```