МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №5

По курсу «Вычислительные методы алгебры»

Метод Данилевского

Вариант №5

Работу выполнил:
студент 3 курса 7 группы
Шатерник Артём
Преподаватель:
Будник А. М.

Минск 2023

1. Постановка задачи.

Дана матрица следующего вида

A

0.5757	-0.0758	0.0152	0.0303	0.1061
0.0788	0.9014	0.0000	-0.0606	0.0606
0.0455	0.0000	0.7242	-0.2121	0.1212
-0.0909	0.1909	0.0000	0.7121	-0.0303
0.3788	0.0000	0.1364	0.0152	0.8484

Требуется методом Данилевского построить собственный многочлен $P_n(\lambda)$ матрицы $A^T A$. Вычислить невязки $\varphi_i(\lambda_i) = P_n(\lambda_i)$ и оценить их значения.

2. Алгоритм решения.

Матрица $A^T A$ имеет следующий вид

0.49146	0.01004	0.09337	- 0.05595	0.3955
0.01004	0.85471	- 0.00115	0.07902	0.0408
0.09337	- 0.00115	0.5433	- 0.15107	0.20511
- 0.05595	0.07902	- 0.15107	0.55689	- 0.03484
0.3955	0.0408	0.20511	- 0.03484	0.75032

Метод Данилевского основан на приведении матрицы к виду Фробениуса, при таком виде сразу видны коэффициенты собственного многочлена этой матрицы. Для приведения к такому виду применяются преобразования подобия $A_i = M_{i-1}^{-1}\, A_{i-1}\, M_{i-1}$. Матрицы M_{i-1} , M_{i-1}^{-1} строятся следующим образом

$$M_{n-1} = \begin{pmatrix} 1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ -\dfrac{a_{n1}}{a_{nn-1}} & \dots & \dfrac{1}{a_{nn-1}} & -\dfrac{a_{nn}}{a_{nn-1}} \\ 0 & \dots & 0 & 1 \end{pmatrix} \quad M_{nn-1}^{-1} = \begin{pmatrix} 1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ a_{n1} & \dots & a_{nn-1} & a_{nn} \\ 0 & \dots & 0 & 1 \end{pmatrix}$$

Обозначим $B = A_{i-1}M_{i-1}$, тогда

$$b_{ij} = a_{ij} + a_{in-1}m_{n-1j} \quad ,j \neq n-1$$

$$b_{in-1} = a_{in-1}m_{n-1n-1} \quad ,j = n-1 \qquad i = \overline{1,n}$$

и

В таком случае $A_i = M_{i-1}^{-1} A_{i-1} M_{i-1}$ вычисляется следующим образом

$$a_{ij}^1 = b_{ij} \qquad ,i = \overline{1, n-1} \ ,j = \overline{1,n}$$

$$a_{n-1j}^1 = \sum_{k=1}^{n} a_{nk} b_{kj} \qquad ,j = \overline{1,n}.$$

Всего потребуется $n-1$ итерация.

Элементы первой строки и будут коэффициентами собственного многочлена матрицы.

3. Листинг программы.

```
import numpy as np
import math

size = 5
a_matrix = []
with open('input.txt') as file:
    i = 0
    for line in file:
        a_matrix.append([float(x) for x in line.split(' ')])
        i += 1
a_matrix = np.array(a_matrix)

# Симметричный вид
a_matrix = np.matmul(a_matrix.T, a_matrix)
for i in range(size):
    for j in range(size):
        print(np.round(a_matrix[i][j], 5), end='')
        print(" ", end='')
    print()
print(np.trace(a_matrix))
# Метод Данилевского
for i in reversed(range(size - 1)):
    m_vec = []
    for j in range(size):
        if j != i:
            m_vec.append(-1 * a_matrix[i + 1][j] / a_matrix[i + 1][i])
        else:
            m_vec.append(1 / a_matrix[i + 1][i])
    m_inv_vec = np.identity(size)
    m_inv_vec[i] = a_matrix[i + 1]
# Умножаем на M справа
    b_matrix = np.zeros((size, size))
    for n in range(size):
       for k in range(size):
           if k != i:
                b_matrix[n][k] = a_matrix[n][k] + a_matrix[n][i] *
m_vec[k]
           else:
               b_matrix[n][k] = a_matrix[n][i] * m_vec[i]
    a_matrix = np.copy(b_matrix)
# Умножаем на M^-1 слева
    for j in range(size):
```

```
        sum = 0
        for k in range(size):
            sum += a_matrix[k][j] * m_inv_vec[i][k]
        b_matrix[i][j] = sum
    a_matrix = np.copy(b_matrix)
print(a_matrix)
print(a_matrix[0])

# Невязки
p = [3.1966884499999972, -3.7968475734971836, 2.0678062361750578, -
0.5082483413019262, 0.044096040836178144]
for i in range(size):
    sum = pow(a_matrix[i][i], size)
    for j in reversed(range(size)):
        sum -= pow(a_matrix[i][i], j) * p[size - j - 1]
    print(sum)
```

4. Результат и его анализ.

Первая строка полученной матрицы имеет вид

3.19668	-3.79684	2.06780	-0.50824	0.04409

Эти числа являются коэффициентами собственного многочлена

$$P(\lambda) = \lambda^5 - p_1\lambda^4 - \dots - p_4\lambda - p_5.$$

То есть это p_i.

Возьмём собственные значения матрицы, подсчитанные с помощью метода вращений с точностью 16 знаков и найдём невязки вида:

$$\varphi_i(\lambda_i) = P_n(\lambda_i).$$

Получим

8.326672684688674e-17
2.636779683484747e-16
6.938893903907228e-17
9.71445146547012e-17
-1.249000902703301e-16

Экономичность:

Метод имеет сложность $O(n^3)$. Это говорит о том, что его использование при больших размерностях матриц затруднительно.
Также проблемой являются нерегулярные случаи, которые увеличивают число операций, но в данной работе не рассматривались.

Точность:

Невязки для собственных значений имеют 16 и 17 порядок. Это говорит о том, что метод является точным, приближённые значения мы получаем только из-за числа хранимых знаков у переменных типа float в Python (17 знаков) и связанных с этим округлений.