**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Факультет прикладной математики и информатики

Лабораторная работа №8 по курсу “ВМА”

“Степенной метод”

Вариант №3

Выполнил: Ёда Никита

3 курс, 6(а) группа

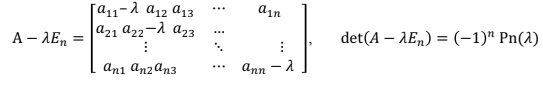
Преподаватель: Будник А.М.

2023

**Постановка задачи**

Необходимо найти собственные значения и собственные векторы

матрицы А:



С помощью построения итерационной последовательности :

**Алгоритм решения**

Степенной метод является итерационным методом решения полной проблемы собственных значений.

Суть метода заключается в последовательном приближении к собственному вектору соответствующему максимальному собственному значению . За берётся отношение соответствующий произвольных координат векторов . Итерационный процесс останавливается, когда:

||

Возьмём начальное приближение:

а последующее будем вычислять как:

,

за x можно принять x ≈

**Листинг программы**

import numpy as np

from sympy import Symbol, solve

A = [[0.3857, -0.0508, 0.0102, 0.0203, 0.0711],

    [0.0528, 0.6039, 0.0000, -0.0406, 0.0406],

    [0.0305, 0.0000, 0.4852, -0.1421, 0.0812],

    [-0.0609, 0.1279, 0.0000, 0.4711, -0.0203],

    [0.2538, 0.0000, 0.0914, 0.0102, 0.5684]]

n = 5

eps = 10 \*\* (-15)

a = np.array(A)

At = a.transpose()

print("Симметрическая A\*A^T:\n\t", At)

a = np.dot(At, a)

yk = np.ones(5)

y = np.dot(a, yk)

l = y[0] / yk[0]

k = 1

while (True):

    yk = np.dot(a, y)

    lk = yk[0] / y[0]

    yk /= max(yk)

    if abs(lk - l) <= eps:

        break

    y = yk

    l = lk

    k += 1

p = [1.42919690e+00, -7.58323068e-01, 1.84357781e-01, -2.02232134e-02, 7.83880398e-04]

print("\nКоэффициенты собственного многочлена P(lambda):\n\t", p)

x = Symbol('x')

Lambda = solve(x\*\*5 - p[0] \* x\*\*4 - p[1] \* x\*\*3 - p[2] \* x\*\*2 - p[3] \* x - p[4], x)

print("\nСобственные значения lambda:\n\t", Lambda)

l = max(Lambda)

print("\nМаксимальное собственное lambdaMax:\n\t" , l)

print("\nКоличество итераций k:\n\t", k)

r = np.dot(a, yk) - lk \* yk

print("\nВектор невязки r:\n\t", r)

print("\nЭпсилон E:\n\t", eps)

p.insert(0, -1)

r1 = sum(-(lk \*\* (n - i)) \* p[i] for i in range(n + 1))

rnorm = np.linalg.norm(r, 1)

print("\nНорма невязки ||r||:\n\t", rnorm)

**Вывод**

Симметрическая A\*A^T:

[[ 0.3857 0.0528 0.0305 -0.0609 0.2538]

[-0.0508 0.6039 0. 0.1279 0. ]

[ 0.0102 0. 0.4852 0. 0.0914]

[ 0.0203 -0.0406 -0.1421 0.4711 0.0102]

[ 0.0711 0.0406 0.0812 -0.0203 0.5684]]

Коэффициенты собственного многочлена P(lambda):

[1.4291969, -0.758323068, 0.184357781, -0.0202232134, 0.000783880398]

Собственные значения lambda:

[0.0856647767244302, 0.170118021950960, 0.265810047188250, 0.394004423114266, 0.513599631022094]

Максимальное собственное lambdaMax:

0.513599631022094

Количество итераций k:

101

Вектор невязки r:

[-6.10622664e-16 -2.06792916e-13 2.17048601e-14 -5.72736303e-14 -1.01030295e-14]

Эпсилон E:

1e-15

Невязка Pn(lambda^k):

6.674626129576922e-11

Норма невязки ||r||:

2.9648505872614805e-13

**Анализ**

С помощью степенного метода мы нашли максимальное по модулю собственное значение и соответствующий ему собственный с точностью порядка 10-13 за 101 итерацию для эпсилона порядка 10-15. Невязка собственного многочлена также довльно близка к нулю (порядка 10-8), что означает, что собственное значение также найдено правильно. Собственное значение и собственный вектор также совпадают с получеными ранее методами Крылова и Данилевского. Чтобы степенной метод сходился необходимо и достаточно, чтобы у матрицы были доминирующее собственное значение.