Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 4.1

по курсу «Численные методы линейной алгебры»

«Вычисление собственных значений и собственных векторов симметричной матрицы методом А.М. Данилевского»

Студентка группы ИУ9-72Б Самохвалова П. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Цель работы

Реализовать метод вычисления собственных значений и собственных векторов симметричной матрицы методом А.М. Данилевского.

2 Задание

- Реализовать метод поиска собственных значений действительной симметричной матрицы A размером 4x4.
- Проверить корректность вычисления собственных значений по теореме Виета.
- Проверить выполнение условий теоремы Гершгорина о принадлежности собственных значений соответствующим объединениям кругов Гершгорина.
- Вычислить собственные вектора и проверить выполнение условия ортогональности собственных векторов.
- Проверить решение на матрице приведенной в презентации.
- Продемонстрировать работу приложения для произвольных симметричных матриц размером n x n с учетом выполнения пунктов приведенных выше.

3 Практическая реализация

Исходный код программы представлен в листинге 1.

Листинг 1: Вычисление собственных значений и собственных векторов симметричной матрицы методом А.М. Данилевского

```
import copy
import random

from num_methods import *


```

```
def func(p):
    n = len(p)
    p = [1] + p
    return lambda x: (-1) ** n * sum([x ** i * p[n - i] * (-1 if i == n else 1)
                                      for i in range(n, -1, -1)])


def div_half_method(a, b, f):
    s = 0.1
    d = 0.0001
    res = []
    x_last = a
    x = x_last
    while x <= b:
        x = x_last + s
        if f(x) * f(x_last) < 0:
            x_left = x_last
            x_right = x
            x_mid = (x + x_last) / 2
            while abs(f(x_mid)) >= d:
                if f(x_left) * f(x_mid) < 0:
                    x_right = x_mid
                else:
                    x_left = x_mid
                x_mid = (x_left + x_right) / 2
            res.append(x_mid)
        x_last = x
    return res


def gershgorin_rounds(a):
    left = -100000
    right = 100000
    n = len(a)
    for i in range(n):
        s = 0
        for j in range(n):
            if i != j:
                s += abs(a[i][i])
        b1 = a[i][i] - s
        b2 = a[i][i] + s
        if i == 0:
            left = b1
            right = b2
        elif b1 < left:
```

```
            left = b1
        elif b2 > right:
            right = b2
    return [left, right]


def danilevsky_method(a):
    n = len(a)
    m = n - 1

    b = [[0] * n for i in range(n)]
    for i in range(n):
        b[i][i] = 1
    for j in range(n):
        if j != m - 1:
            b[m - 1][j] = -a[m][j] / a[m][m - 1]
    b[m - 1][m - 1] = 1 / a[m][m - 1]

    b_mul = copy.deepcopy(b)

    c = [[0] * n for i in range(n)]
    for i in range(n):
        c[i][m - 1] = a[i][m - 1] * b[m - 1][m - 1]
    for i in range(n - 1):
        for j in range(n):
            if j != m - 1:
                c[i][j] = a[i][j] + a[i][m - 1] * b[m - 1][j]

    b_inv = [[0] * n for i in range(n)]
    for i in range(n):
        b_inv[i][i] = 1
    for j in range(n):
        b_inv[m - 1][j] = a[m][j]

    d = [[0] * n for i in range(n)]
    for i in range(m - 1):
        for j in range(n):
            d[i][j] = c[i][j]
    for j in range(n):
        for k in range(n):
            d[m - 1][j] += a[m][k] * c[k][j]
    d[m][m - 1] = 1

    for k in range(2, n):
        b = [[0] * n for i in range(n)]
        for i in range(n):
```

```python
                b[i][i] = 1
            for j in range(n):
                if j != m - k:
                    b[m - k][j] = -d[m - k + 1][j] / d[m - k + 1][m - k]
            b[m - k][m - k] = 1 / d[m - k + 1][m - k]

            b_mul = mult_matr_matr(b_mul, b)

            b_inv = inv_matr(b)
            d = mult_matr_matr(b_inv, d)
            d = mult_matr_matr(d, b)
    return d, b_mul


def generate_symm_matrix(n, v1, v2):
    a = [[0] * n for i in range(n)]
    for i in range(n):
        for j in range(i, n):
            a[i][j] = random.uniform(v1, v2)
            if i != j:
                a[j][i] = a[i][j]
    return a


def check_ortonormal(vs):
    for v in vs:
        if abs(norm_vec(v) - 1) > 0.001:
            return False
    n = len(vs)
    for i in range(n):
        for j in range(i + 1, n):
            if abs(scalar_mult_vec(vs[i], vs[j])) > 0.001:
                return False
    return True


# n = 7
# a = generate_symm_matrix(n, -10, 10)

n = 4
a = [[2.2, 1, 0.5, 2],
     [1, 1.3, 2, 1],
     [0.5, 2, 0.5, 1.6],
     [2, 1, 1.6, 2]]

d, b = danilevsky_method(a)
```

```python
p = d[0][:]

g = gershgorin_rounds(a)
print("Boundaries of search for roots from Gershgorin's theorem")
print(g)
print()

ls = div_half_method(g[0], g[1], func(p))
print("Eigenvalues of matrix")
print(ls)
print()

print("Checking Viet theorem")
s = 0
for i in range(len(ls)):
    s += ls[i]
m = 1
for i in range(len(ls)):
    m *= ls[i]
if abs(s - trace_matr(a)) < 0.001 and abs(m - det_matr(a)) < 0.001:
    print("Eigenvalues of matrix satisfy Viet theorem")
else:
    print("Eigenvalues of matrix do not satisfy Viet theorem")
print()

print("Eigenvectors of matrix")
vectors = []
for l in ls:
    y = [1]
    for i in range(1, n):
        y.append(l ** i)
    y = y[::-1]
    y = mult_matr_vec(b, y)
    norm = norm_vec(y)
    for i in range(n):
        y[i] /= norm
    vectors.append(y)
    print(y)
print()

print("Checking for orthonormality")
if check_ortonormal(vectors):
    print("Vectors are orthonormal")
else:
    print("Vectors are not orthonormal")
```

4 Результаты

Результаты работы программы представлены на рисунках 1 – 5.

```
Boundaries of search for roots from Gershgorin's theorem
[-4.4, 8.8]
```

Рис. 1 — Границы поиска корней из теоремы Гершгорина

```
Eigenvalues of matrix
[-1.4200866699218744, 0.22263183593750066, 1.545422363281251, 5.652032470703123]
```

Рис. 2 — Собственные значения матрицы

```
Checking Viet theorem
Eigenvalues of matrix satisfy Viet theorem
```

Рис. 3 — Проверка собственных значений по теореме Виета

```
Eigenvectors of matrix
[-0.2220427470401971‌4, 0.5159103003083314, -0.7572742854728591, 0.33327051637398253]
[-0.5219259121089701, -0.4548638305168229, 0.15344952193407355, 0.7050854431733695]
[0.6289303187008084, -0.5725734703768309, -0.48565425679175006, 0.2018569248244534]
[0.5317353434587488, 0.44619446527385787, 0.4088161714145183, 0.5924840602630231]
```

Рис. 4 — Собственные векторы матрицы

```
Checking for orthonormality
Vectors are orthonormal
```

Рис. 5 — Проверка ортонормированности собственных векторов

5 Выводы

В результате выполнения лабораторной работы был реализован метод вычисления собственных значений и собственных векторов симметричной матрицы методом А.М. Данилевского.