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

по курсу «Численные методы линейной алгебры»

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

Студент группы ИУ9-71Б Баев Д.А

Преподаватель Посевин Д. П.

Москва 2023

1 Задание

1. Реализовать метод поиска собственных значений действительной симметричной матрицы A размером 4x4.

2. Проверить корректность вычисления собственных значений по теореме Виета.

3. Проверить выполнение условий теоремы Гершгорина о принадлежности собственных значений соответствующим объединениям кругов Гершгорина.

4. Вычислить собственные вектора и проверить выполнение условия ортогональности собственных векторов.

5. Проверить решение на матрице приведенной в презентации.

6. Продемонстрировать работу приложения для произвольных симметричных матриц размером n x n с учетом выполнения пунктов приведенных выше.

2 Исходный код

Исходный код программы представлен в листингах 1– 4.

Листинг 1 — Вычисление собственных значений и векторов со всеми проверками

```
def eig(A):
    assert len(A) == len(A[0])
    assert is_sym_matrix(A)
    n = len(A)
    intervals = gerchgoin_intervals(A)
    trace = sum(A[i][i] for i in range(n))
    coefs, B = danilevskiy(A)
    coefs = list(map(lambda x: x * -1, coefs))
    f = lambda x: np.polyval([1] + coefs, x)
    search_intervals = binary_search_intervals(intervals, f)
    eigs = binary_search_roots(search_intervals, f)
    check_gerchgoin(intervals, eigs)
    print(f"Viet theorem:\nSum = {sum(eigs)}\nTrace = {trace}")
    eig_vectors = []
    for eig in eigs:
        y_vector = [eig ** i for i in range(n - 1, -1, -1)]
        x_vector = np.array(mul_matrix_by_vector(B, y_vector))
        eig_vectors.append(x_vector / norm(x_vector))
    print("Ortnorm eig vectors:")
    print('Norms:')
    for vector in eig_vectors:
        print(norm(vector))
    print("Scalar prods:")
    for i in range(n - 1):
        for j in range(i + 1, n):
            print(scalar(eig_vectors[i], eig_vectors[j]))
    return np.array(eigs), eig_vectors
```

Листинг 2 — Вычисление матрицы Фробениуса и матрицы B методом А.М. Данилевского

```
def danilevskiy(A):
    n = len(A)
    B_i = np.diag(np.ones(n))
    for i in range(n):
        if i != (n - 2):
            B_i[n - 2][i] = -1 * A[n - 1][i] / A[n - 1][n - 2]
        else:
            B_i[n - 2][n - 2] =  1 / A[n - 1][n - 2]
    B_i_inv = np.diag(np.ones(n))
    B_i_inv[n - 2] = deepcopy(A[n - 1])
    P = mul_matrix(B_i_inv, A)
    P = mul_matrix(P, B_i)
    B = deepcopy(B_i)
    for i in range(n - 3, -1, -1):
        B_i = np.diag(np.ones(n))
        for j in range(n):
            if j != i:
                B_i[i][j] = -1 * P[i + 1][j] / P[i + 1][i]
            else:
                B_i[i][i] = 1 / P[i + 1][i]
        B_i_inv = np.diag(np.ones(n))
        B_i_inv[i] = deepcopy(P[i + 1])
        P = mul_matrix(B_i_inv, P)
        P = mul_matrix(P, B_i)
        B = mul_matrix(B, B_i)

    return P[0], B
```

Листинг 3 — Вычисление и объединение кругов Гершгорина

```
def gerchgoin_intervals(A):
    intervals = []
    for i in range(len(A)):
        center = A[i][i]
        radius = sum(abs(A[i][j]) for j in range(len(A)) if j != i)
        intervals.append([center - radius, center + radius])

    intervals.sort(key=lambda x:x[0])
    merged = [intervals[0]]
    for i in range(1, len(intervals)):
        current_interval = intervals[i]
        previous_interval = merged[-1]
        if current_interval[0] <= previous_interval[1]:
            merged[-1] = [previous_interval[0], max(previous_interval
    [1], current_interval[1])]
        else:
            merged.append(current_interval)

    return merged

def check_gerchgoin(intervals, eigs):
    for eig in eigs:
        interval_found = False
        for interval in intervals:
            if interval[0] <= eig <= interval[1]:
                interval_found = True
                break
        if not interval_found:
            print(f"Eig: {eig} not in gerchgoin circles")
            return
    print("All eigs in gerchgoin circles")
```

Листинг 4 — Решение характеристического уравнения методом бинарного поиска

```
def binary_search_roots(intervals, f, delta=1e-3):
    lambdas = []
    for interval in intervals:
        left = interval[0]
        right = interval[1]
        f_left = f(left)
        assert f_left * f(right) < 0
        x = (left + right) / 2
        f_x = f(x)
        while abs(f_x) > delta:
            if f_left * f_x < 0:
                right = x
            else:
                left = x
                f_left = f(left)
            x = (left + right) / 2
            if x == left or x == right:
                break
            f_x = f(x)
        lambdas.append(x)
    return lambdas

def binary_search_intervals(intervals, f, delta=0.1):
    search_intervals = []
    for interval in intervals:
        left = interval[0]
        right = interval[1]
        delta_x = left + delta
        while delta_x <= right:
            f_left = f(left)
            f_delta = f(delta_x)
            if f_left * f_delta < 0:
                search_intervals.append([left, delta_x])
                left = delta_x
                delta_x = left + delta
            else:
                delta_x += delta
    return search_intervals
```

3 Результаты

Результат поиска собственных значений и векторов и все необходимые проверки представлены на рисунках 1- 3.

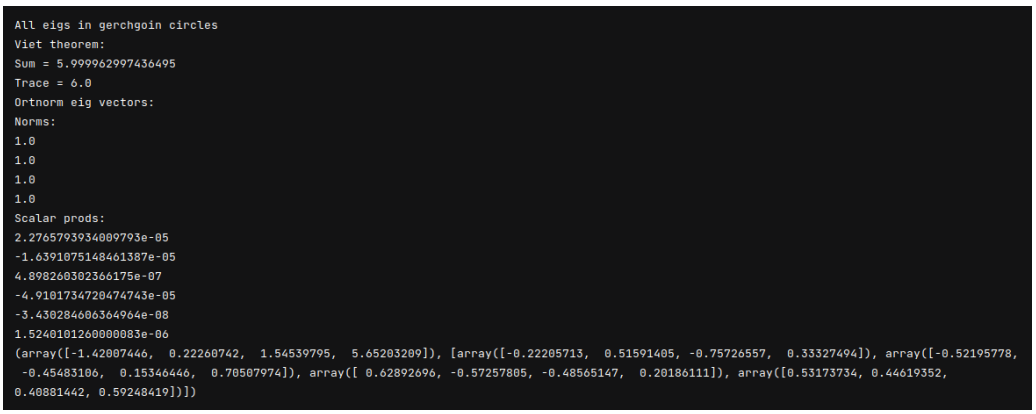

Рис. 1 — Результат поиска собственных значений и векторов для матрицы 4x4 (из презентации)

Рис. 2 — Результат поиска собственных значений и векторов для произвольной матрицы 10x10

Рис. 3 — Результат поиска собственных значений и векторов для произвольной матрицы 30x30

4 Выводы

В результате выполнения лабораторной работы был реализован метод для поиска собственных значений и нормированной системы собственных векторов произвольной квадратной действительной симметричной матрицы. Для поиска собственных значений метод использует метод бинарного поиска, локализуя область поиска с помощью теоремы Гершгорина. Коофициенты характеристического уравнения и собственные вектора вычисляются при помощи метода А.М. Данилевского.