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 № 4.2

по курсу «Численные методы линейной алгебры»

«Вычисление собственных значений и собственных векторов симметричной матрицы методом А.Н. Крылова»

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 коофициентов характеристического уравнения и собственных векторов методом А.Н. Крылова

```
    def krylov_eig_vectors(A, Y, coefs, eigs):
        n = len(A)
        Q = np.zeros(shape=(n, n))
        for i in range(n):
            for j in range(n):
                Q[j][i] = 1 if j == 0 else eigs[i] * Q[j - 1][i] + coefs[
    j - 1]
        vectors = []
        for i in range(n):
            vector = deepcopy(Y[0])
            for j in range(1, n):
                vector += Q[j][i] * Y[j]
            vectors.append(vector / norm(vector))
        return vectors


    def krylov(A):
        n = len(A)
        y = np.random.uniform(-10, 10, n)
        Y = [y]
        for i in range(n - 1):
            y = mul_matrix_by_vector(A, y)
            Y.append(y)
        b = mul_matrix_by_vector(A, y)
        Y = np.array(list(reversed(Y)))
        P = Y.T
        coefs = np.linalg.solve(P, b)
        return coefs, Y

```

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.

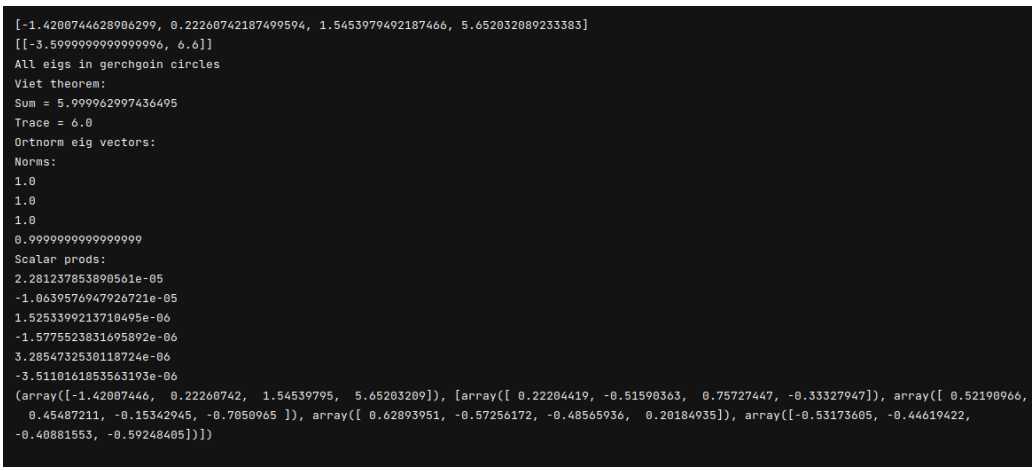

Рис. 1 — Результат поиска собственных значений и векторов для матрицы 4x4 (из презентации)

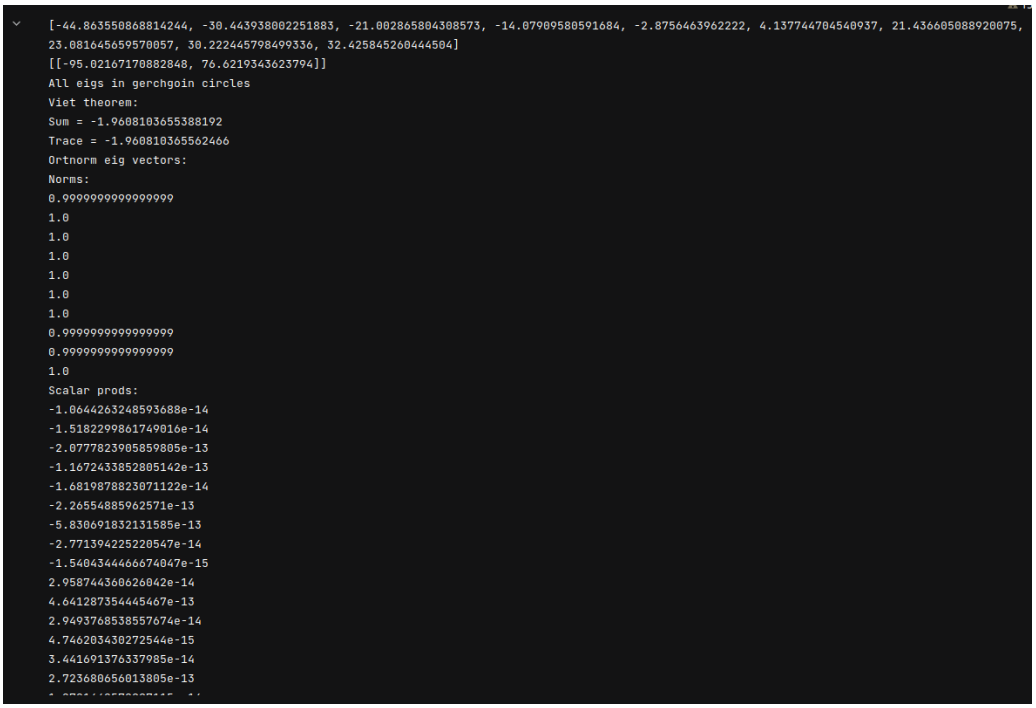

Рис. 2 — Результат поиска собственных значений и векторов для произвольной матрицы 10x10

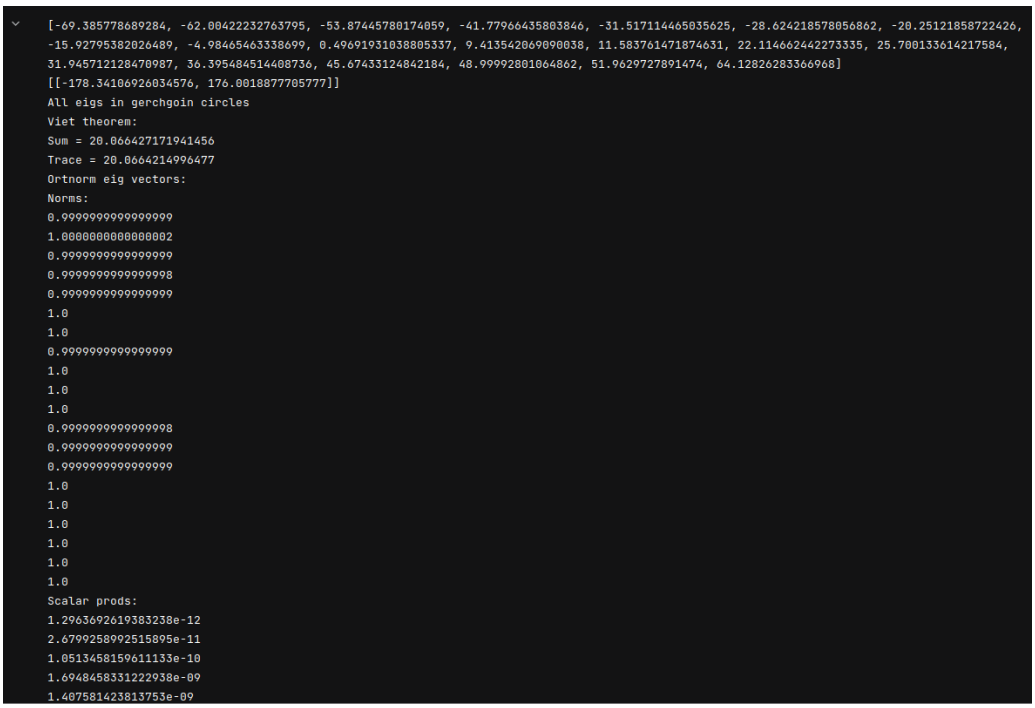

Рис. 3 — Результат поиска собственных значений и векторов для произвольной матрицы 20x20

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.Н. Крылова.