



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 4×4 .
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 $n \times n$ 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

```
1 def eig(A):
2     assert len(A) == len(A[0])
3     assert is_sym_matrix(A)
4     n = len(A)
5     intervals = gerchgoi_intervals(A)
6     trace = sum(A[i][i] for i in range(n))
7     coefs, B = danilevskiy(A)
8     coefs = list(map(lambda x: x * -1, coefs))
9     f = lambda x: np.polyval([1] + coefs, x)
10    search_intervals = binary_search_intervals(intervals, f)
11    eigs = binary_search_roots(search_intervals, f)
12    check_gerschgoi(intervals, eigs)
13    print(f"Viet theorem:\nSum = {sum(eigs)}\nTrace = {trace}")
14    eig_vectors = []
15    for eig in eigs:
16        y_vector = [eig ** i for i in range(n - 1, -1, -1)]
17        x_vector = np.array(mul_matrix_by_vector(B, y_vector))
18        eig_vectors.append(x_vector / norm(x_vector))
19    print("Ortnorm eig vectors:")
20    print('Norms:')
21    for vector in eig_vectors:
22        print(norm(vector))
23    print("Scalar prods:")
24    for i in range(n - 1):
25        for j in range(i + 1, n):
26            print(scalar(eig_vectors[i], eig_vectors[j]))
27    return np.array(eigs), eig_vectors
```

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

```
1 def danilevskiy(A):
2     n = len(A)
3     B_i = np.diag(np.ones(n))
4     for i in range(n):
5         if i != (n - 2):
6             B_i[n - 2][i] = -1 * A[n - 1][i] / A[n - 1][n - 2]
7         else:
8             B_i[n - 2][n - 2] = 1 / A[n - 1][n - 2]
9     B_i_inv = np.diag(np.ones(n))
10    B_i_inv[n - 2] = deepcopy(A[n - 1])
11    P = mul_matrix(B_i_inv, A)
12    P = mul_matrix(P, B_i)
13    B = deepcopy(B_i)
14    for i in range(n - 3, -1, -1):
15        B_i = np.diag(np.ones(n))
16        for j in range(n):
17            if j != i:
18                B_i[i][j] = -1 * P[i + 1][j] / P[i + 1][i]
19            else:
20                B_i[i][i] = 1 / P[i + 1][i]
21        B_i_inv = np.diag(np.ones(n))
22        B_i_inv[i] = deepcopy(P[i + 1])
23        P = mul_matrix(B_i_inv, P)
24        P = mul_matrix(P, B_i)
25        B = mul_matrix(B, B_i)
26
27    return P[0], B
```

Листинг 3 — Вычисление и объединение кругов Гершгорина

```
1 def gerchgoин_intervals(A):
2     intervals = []
3     for i in range(len(A)):
4         center = A[i][i]
5         radius = sum(abs(A[i][j]) for j in range(len(A)) if j != i)
6         intervals.append([center - radius, center + radius])
7
8     intervals.sort(key=lambda x:x[0])
9     merged = [intervals[0]]
10    for i in range(1, len(intervals)):
11        current_interval = intervals[i]
12        previous_interval = merged[-1]
13        if current_interval[0] <= previous_interval[1]:
14            merged[-1] = [previous_interval[0], max(previous_interval
15            [1], current_interval[1])]
16        else:
17            merged.append(current_interval)
18
19    return merged
20
21 def check_gerchgoин(intervals, eigs):
22     for eig in eigs:
23         interval_found = False
24         for interval in intervals:
25             if interval[0] <= eig <= interval[1]:
26                 interval_found = True
27                 break
28         if not interval_found:
29             print(f"Eig: {eig} not in gerchgoин circles")
30     print("All eigs in gerchgoин circles")
```

Листинг 4 — Решение характеристического уравнения методом бинарного поиска

```
1 def binary_search_roots(intervals, f, delta=1e-3):
2     lambdas = []
3     for interval in intervals:
4         left = interval[0]
5         right = interval[1]
6         f_left = f(left)
7         assert f_left * f(right) < 0
8         x = (left + right) / 2
9         f_x = f(x)
10        while abs(f_x) > delta:
11            if f_left * f_x < 0:
12                right = x
13            else:
14                left = x
15                f_left = f(left)
16            x = (left + right) / 2
17            if x == left or x == right:
18                break
19            f_x = f(x)
20        lambdas.append(x)
21    return lambdas
22
23 def binary_search_intervals(intervals, f, delta=0.1):
24     search_intervals = []
25     for interval in intervals:
26         left = interval[0]
27         right = interval[1]
28         delta_x = left + delta
29         while delta_x <= right:
30             f_left = f(left)
31             f_delta = f(delta_x)
32             if f_left * f_delta < 0:
33                 search_intervals.append([left, delta_x])
34                 left = delta_x
35                 delta_x = left + delta
36             else:
37                 delta_x += delta
38    return search_intervals
```

3 Результаты

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

```

All eigs in gerschgorin circles
Viet theorem:
Sum = 5.999962997436495
Trace = 6.0
Ortnorm eig vectors:
Norms:
1.0
1.0
1.0
1.0
Scalar prods:
2.2765793934889793e-05
-1.6391075148461387e-05
4.898268392366175e-07
-4.9101734728474743e-05
-3.430284686364964e-08
1.524010126000083e-06
(array([-1.42007446,  0.22260742,  1.54539795,  5.65203209]), array([-0.22285713,  0.51591405, -0.75726557,  0.33327494]), array([-0.52195778,
-0.45483106,  0.15346446,  0.70587974]), array([ 0.62892696, -0.57257805, -0.48565147,  0.20186111]), array([0.53173734, 0.44619352,
0.40881442, 0.59248419]]))

```

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

```

All eigs in gerschgorin circles
Viet theorem:
Sum = 2.8880590999130646
Trace = 2.8880590999102616
Ortnorm eig vectors:
Norms:
1.0
0.9999999999999999
1.0
1.0
1.0
1.0
0.9999999999999999
1.0
1.0
0.9999999999999999
1.0
0.9999999999999999
Scalar prods:
-3.1973729219814118e-12
2.9559688838644793e-13
9.623987370921405e-12
-8.814893259767587e-13
6.258181473840825e-12
-2.482763994393622e-12
5.448419493347956e-14
-3.1442983836165215e-13
3.329225783943457e-12
3.640143741989732e-14
-2.558604370039852e-13

```

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

```

All eigs in gerchgoi circles
Viet theorem:
Sum = -46.665709055037155
Trace = -46.6657090558255
Ortnorm eig vectors:
Norms:
1.0000000000000002
0.9999999999999998
0.9999999999999998
1.0
1.0
0.9999999999999999
1.0
1.0
1.0
0.9999999999999999
1.0000000000000002
0.9999999999999998
1.0
1.0
0.9999999999999999
1.0
0.9999999999999999
1.0
1.0
1.0
1.0

```

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. Коэффициенты характеристического уравнения и собственные вектора вычисляются при помощи метода А.М. Данилевского.