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»

Летучка № 2

по курсу «Численные методы линейной алгебры»

«Поиск собственных значений и собственных векторов матриц размером 2x2 и 3x3»

Студентка группы ИУ9-72Б Самохвалова П. С.

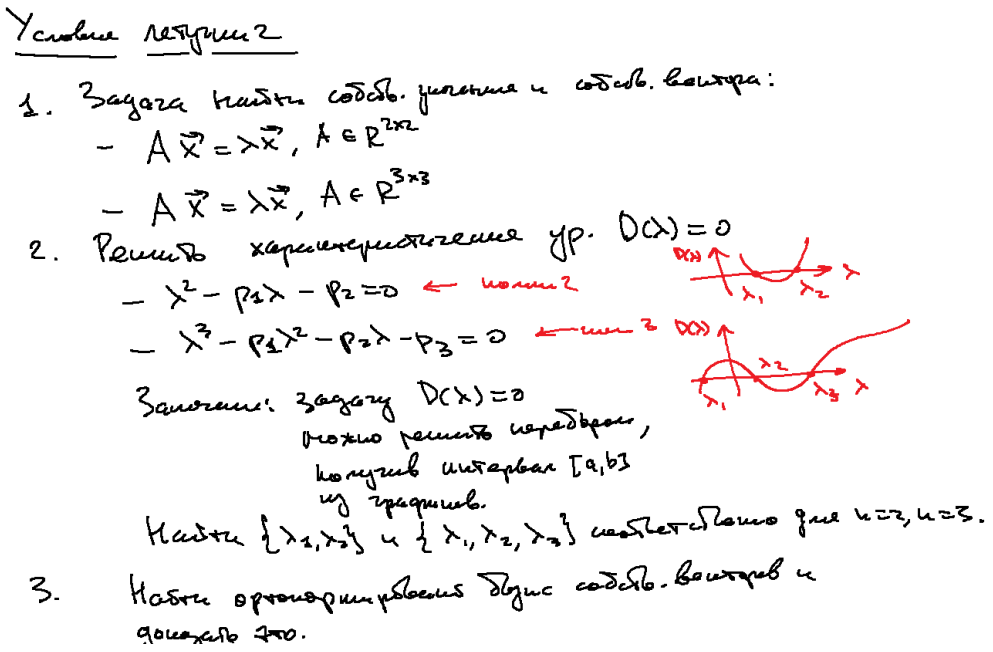
Преподаватель Посевин Д. П.

Москва 2023

1 Цель

Найти собственные значения и собственные векторы матриц размером 2x2 и 3x3.

2 Задание



3 Практическая реализация

Исходный код программы представлен в листинге 1.

Листинг 1: Поиск собственных значений и собственных векторов матриц 2x2 и 3x3

```
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 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:
```

```
57                  c[i][j] = a[i][j] + a[i][m - 1] * b[m - 1][j]
58
59      b_inv = [[0] * n for i in range(n)]
60      for i in range(n):
61          b_inv[i][i] = 1
62      for j in range(n):
63          b_inv[m - 1][j] = a[m][j]
64
65      d = [[0] * n for i in range(n)]
66      for i in range(m - 1):
67          for j in range(n):
68              d[i][j] = c[i][j]
69      for j in range(n):
70          for k in range(n):
71              d[m - 1][j] += a[m][k] * c[k][j]
72      d[m][m - 1] = 1
73
74      for k in range(2, n):
75          b = [[0] * n for i in range(n)]
76          for i in range(n):
77              b[i][i] = 1
78          for j in range(n):
79              if j != m - k:
80                  b[m - k][j] = -d[m - k + 1][j] / d[m - k + 1][m - k]
81          b[m - k][m - k] = 1 / d[m - k + 1][m - k]
82
83          b_mul = mult_matr_matr(b_mul, b)
84
85          b_inv = inv_matr(b)
86          d = mult_matr_matr(b_inv, d)
87          d = mult_matr_matr(d, b)
88      return d, b_mul
89
90
91  def generate_symm_matrix(n, v1, v2):
92      a = [[0] * n for i in range(n)]
93      for i in range(n):
94          for j in range(i, n):
95              a[i][j] = random.uniform(v1, v2)
96              if i != j:
97                  a[j][i] = a[i][j]
98      return a
99
100
101 def check_ortonormal(vs):
102     for v in vs:
```

```
103             if abs(norm_vec(v) - 1) > 0.001:
104                 return False
105         n = len(vs)
106         for i in range(n):
107             for j in range(i + 1, n):
108                 if abs(scalar_mult_vec(vs[i], vs[j])) > 0.001:
109                     return False
110         return True
111 
112 
113 n = 2
114 a = generate_symm_matrix(n, -10, 10)
115 
116 d, b = danilevsky_method(a)
117 p = d[0][:]
118 
119 plot_func(func(p), [i for i in range(-30, 30)])
120 
121 g = [-30, 30]
122 
123 print("Matrix 2x2")
124 print()
125 ls = div_half_method(g[0], g[1], func(p))
126 print("Eigenvalues of matrix")
127 print(ls)
128 print()
129 
130 print("Eigenvectors of matrix")
131 vectors = []
132 for l in ls:
133     y = [1]
134     for i in range(1, n):
135         y.append(l ** i)
136     y = y[::-1]
137     y = mult_matr_vec(b, y)
138     norm = norm_vec(y)
139     for i in range(n):
140         y[i] /= norm
141     vectors.append(y)
142     print(y)
143 print()
144 
145 print("Checking for orthonormality")
146 if check_ortonormal(vectors):
147     print("Vectors are orthonormal")
148 else:
```

```
149     print("Vectors are not orthonormal")
150 print()
151
152 n = 3
153 a = generate_symm_matrix(n, -10, 10)
154
155 d, b = danilevsky_method(a)
156 p = d[0][:]
157
158 plot_func(func(p), [i for i in range(-30, 30)])
159
160 g = [-30, 30]
161
162 print("Matrix 3x3")
163 print()
164 ls = div_half_method(g[0], g[1], func(p))
165 print("Eigenvalues of matrix")
166 print(ls)
167 print()
168
169 print("Eigenvectors of matrix")
170 vectors = []
171 for l in ls:
172     y = [1]
173     for i in range(1, n):
174         y.append(l ** i)
175     y = y[::-1]
176     y = mult_matr_vec(b, y)
177     norm = norm_vec(y)
178     for i in range(n):
179         y[i] /= norm
180     vectors.append(y)
181     print(y)
182 print()
183
184 print("Checking for orthonormality")
185 if check_ortonormal(vectors):
186     print("Vectors are orthonormal")
187 else:
188     print("Vectors are not orthonormal")
```

4 Результаты

Результаты работы программы представлены на рисунках 1 – 4.

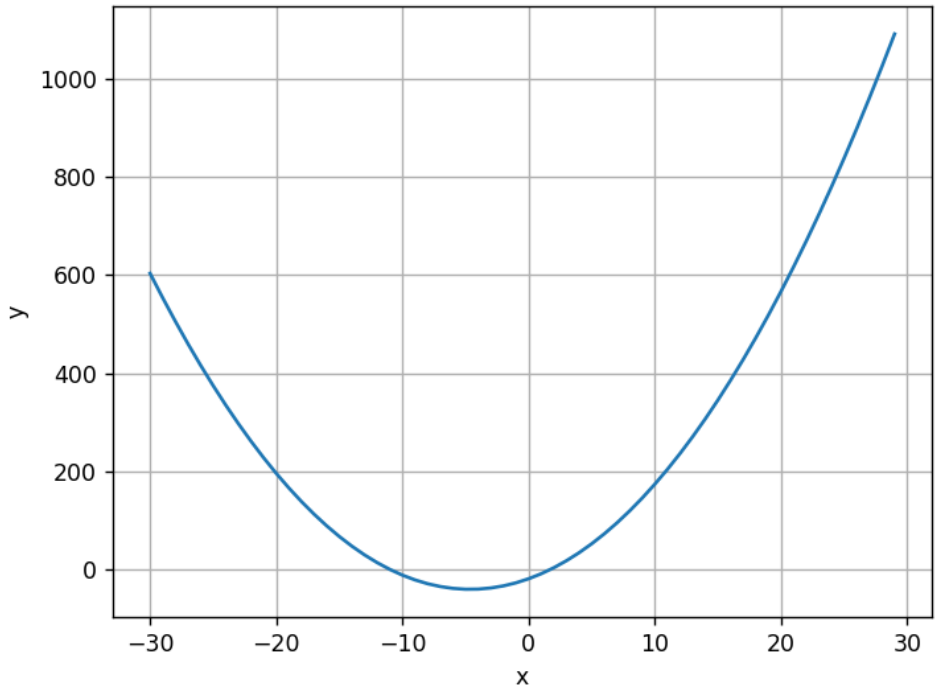


Рис. 1 — График для матрицы 2x2

```
Matrix 2x2

Eigenvalues of matrix
[-10.98774414062482, 1.7213623046876583]

Eigenvectors of matrix
[0.4889666032869862, 0.8723025053672533]
[-0.8723021127286977, 0.4889673037423364]

Checking for orthonormality
Vectors are orthonormal
```

Рис. 2 — Собственные значения и собственные векторы для матрицы 2x2

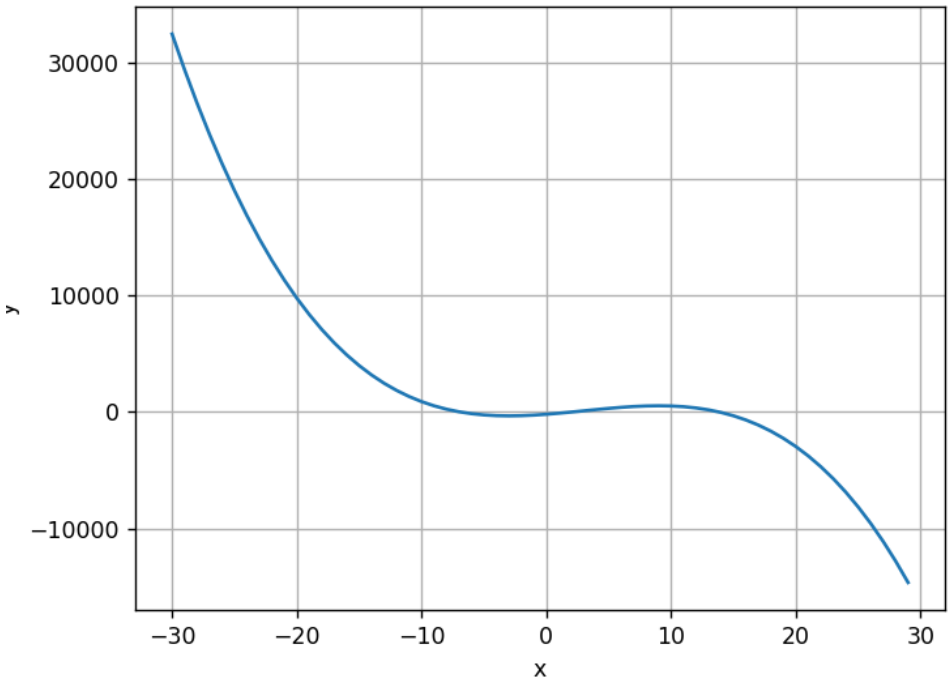


Рис. 3 — График для матрицы 3x3

```
Matrix 3x3

Eigenvalues of matrix
[-6.921556854247883, 2.092974853515783, 13.837934875488408]

Eigenvectors of matrix
[0.1876918379728112, -0.5489024722807847, 0.8145414967221931]
[0.8394363650248318, 0.5202404334696308, 0.15715113889892066]
[-0.510018026124098, 0.6542598042413473, 0.5584135757505837]

Checking for orthonormality
Vectors are orthonormal
```

Рис. 4 — Собственные значения и собственные векторы для матрицы 3x3

5 Выводы

В результате выполнения лабораторной работы были найдены собственные значения и собственные векторы для матриц размером 2x2 и 3x3.