**НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ УКРАИНЫ “КИЕВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ ИМЕНИ ИГОРЯ СИКОРСКОГО”**

**ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ**

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ №4

Дисциплина: «Методы вычислений»

Тема: «Вычисление собственных значений»

Выполнила

студентка 3 курса группы ФИ-41

Лавягина Ольга Алексеевна

Проверила

Стёпочкина Ирина Валерьевна

Киев — 2017 года

ОГЛАВЛЕНИЕ

# 1 ИСХОДНАЯ СИСТЕМА

Нужно найти собственные значения матрицы $A$ с помощью метода Данилевского.

Для варианта 8 матрица имеет вид

$$A = \begin{bmatrix} 5.26 & 0.10 & 0.55 & 1.28 \\ 1.10 & 4.44 & 1.30 & 0.16 \\ 0.55 & 1.30 & 6.44 & 2.10 \\ 1.28 & 0.16 & 2.10 & 8.10 \end{bmatrix},$$

# 2 МАТРИЦЫ $M_I$ И $M_I^{-1}$

```
M3^-1:
[[ 1.    0.    0.    0.  ]
 [ 0.    1.    0.    0.  ]
 [ 1.28  0.16  2.1   8.1 ]
 [ 0.    0.    0.    1.  ]]
M3:
[[ 1.          0.          0.          0.        ]
 [ 0.          1.          0.          0.        ]
 [-0.60952381 -0.07619048  0.47619048 -3.85714286]
 [ 0.          0.          0.          1.        ]]
M2^-1:
[[  1.           0.           0.           0.        ]
 [ -0.73528571   2.46851429  14.97428571 -49.60771429]
 [  0.           0.           1.           0.        ]
 [  0.           0.           0.           1.        ]]
M2:
[[  1.           0.           0.           0.        ]
 [  0.29786569   0.40510197  -6.06611264  20.09618278]
 [  0.           0.           1.           0.        ]
 [  0.           0.           0.           1.        ]]
M1^-1:
[[   0.3173714    19.29793352 -113.01569875  203.12210164]
 [   0.            1.            0.            0.        ]
 [   0.            0.            1.            0.        ]
 [   0.            0.            0.            1.        ]]
M1:
[[   3.15088251  -60.80552127  356.09918898 -640.01387824]
 [   0.            1.            0.            0.        ]
 [   0.            0.            1.            0.        ]
 [   0.            0.            0.            1.        ]]
```

# 3 РЕЗУЛЬТИРУЮЩАЯ МАТРИЦА В ФОРМЕ ФРОБЕНИУСА

```
M1^-1 M2^-1 M3^-1 AM3M2M1:
[[  2.42400000e+01  -2.08379900e+02   7.61624474e+02  -1.00373945e+03]
 [  1.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   1.00000000e+00   0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   1.00000000e+00   0.00000000e+00]]
```

## 4 ПОЛУЧЕННОЕ ХАРАКТЕРИСТИЧЕСКОЕ УРАВНЕНИЕ

Характеристический полином строется раскрытием характеристического определителя матрицы $P = M_1^{-1}M_2^{-1}M_3^{-1}AM_3M_2M_1$ по первой строке. Он, вследствие подобия матриц $A$ и $P$, является так же и характеристическим полиномом матрицы $A$.

Раскрываем по первому столбцу

$$p(\lambda) = (-1)^3 \cdot \left(\lambda^4 - 24.24\lambda^3 + 208.3799\lambda^2 - 761.624474\lambda + 1003.73945\right).$$

# 5 СОБСТВЕННЫЕ ЧИСЛА

Собственные числа были найдены из уравнения $P(\lambda) = 0$. Уравнение имеет 4 действительных корня (рис. 5.1):

$$\lambda_1 = 3.57273,\ \lambda_2 = 4.97342,\ \lambda_3 = 5.59192,\ \lambda_4 = 10.1019.$$

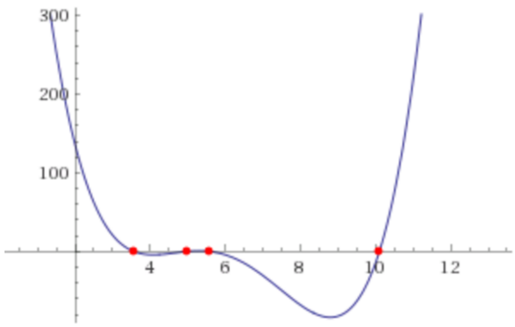


Рисунок 5.1 — График корней

# 6 ЛИСТИНГ ПРОГРАММЫ

## Листинг файла __main__.py

```
from solve import frobenius


if __name__ == '__main__':
    matrix_size = 4
    A = [[5.26, 0.10, 0.55, 1.28],
         [1.10, 4.44, 1.30, 0.16],
         [0.55, 1.30, 6.44, 2.10],
         [1.28, 0.16, 2.10, 8.10]]
    result = frobenius(A)
    #polynomial = danilevsky(result)
```

## Листинг файла solve.py

```
from numpy import array, zeros, round

def frobenius(matrix):
    matrix = array(matrix, dtype='f8', copy=True)
    assert matrix.ndim == 2 and matrix.shape[0] == matrix.shape[1]
    print('A:')
    print(matrix)
    m3 = zeros((4,4), dtype='f8')
    m3[0,0] = 1
    m3[1,1] = 1
    m3[3,3] = 1
    m3[2,0] = matrix[3,0]
    m3[2,1] = matrix[3,1]
    m3[2,2] = matrix[3,2]
    m3[2,3] = matrix[3,3]
    print('M3^-1:')
    print(m3)
    M3 = m3.copy()
    M3[2,0] = -m3[2,0] / matrix[3,2]
    M3[2,1] = -m3[2,1] / matrix[3,2]
    M3[2,2] = 1 / matrix[3,2]
    M3[2,3] = -m3[2,3] / matrix[3,2]
    print('M3:')
    print(M3)
    m3AM3 = m3.dot(matrix.dot(M3))
    print('M3^-1 AM3:')
    print(m3AM3)

    m2 = zeros((4,4), dtype='f8')
    m2[0,0] = 1
    m2[2,2] = 1
```

```
m2[3,3] = 1
m2[1,0] = m3AM3[2,0]
m2[1,1] = m3AM3[2,1]
m2[1,2] = m3AM3[2,2]
m2[1,3] = m3AM3[2,3]
print('M2^-1:')
print(m2)
M2 = m2.copy()
M2[1,0] = -m2[1,0] / m3AM3[2,1]
M2[1,1] = 1 / m3AM3[2,1]
M2[1,2] = -m2[1,2] / m3AM3[2,1]
M2[1,3] = -m2[1,3] / m3AM3[2,1]
print('M2:')
print(M2)
m2m3AM3M2 = round(m2.dot(m3AM3.dot(M2)), 15)
print('M2^-1 M3^-1 AM3M2:')
print(m2m3AM3M2)

m1 = zeros((4,4), dtype='f8')
m1[1,1] = 1
m1[2,2] = 1
m1[3,3] = 1
m1[0,0] = m2m3AM3M2[1,0]
m1[0,1] = m2m3AM3M2[1,1]
m1[0,2] = m2m3AM3M2[1,2]
m1[0,3] = m2m3AM3M2[1,3]
print('M1^-1:')
print(m1)
M1 = m1.copy()
M1[0,0] = 1 / m2m3AM3M2[1,0]
M1[0,1] = - m1[0,1] / m2m3AM3M2[1,0]
M1[0,2] = - m1[0,2] / m2m3AM3M2[1,0]
M1[0,3] = - m1[0,3] / m2m3AM3M2[1,0]
print('M1:')
print(M1)
P = round(m1.dot(m2m3AM3M2.dot(M1)), 8)
print('M1^-1 M2^-1 M3^-1 AM3M2M1:')
print(P)

return P
```

# 7 ПРОВЕРКА ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ В WOLFRAMALPHA

Была использована функция eigenvalues и получены такие результаты (рис. 7.1):

$$\lambda_1 = 10.1019,\ \lambda_2 = 5.59192,\ \lambda_3 = 4.97342,\ \lambda_4 = 2.57273.$$

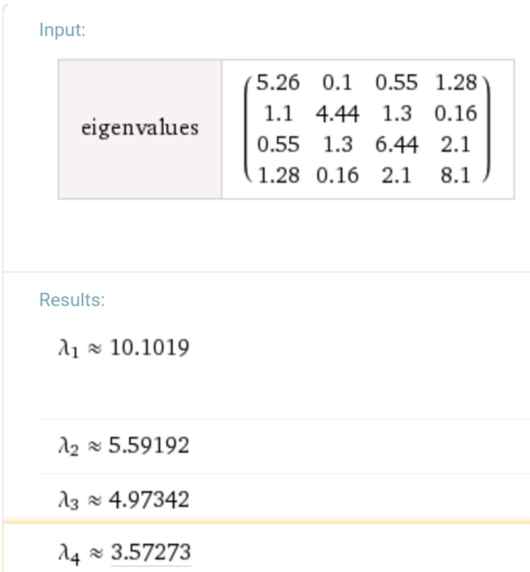


Рисунок 7.1 — Проверка полученных результатов в WolframAlpha

# ВЫВОДЫ

Собственные числа матрицы были найдены с помощью метода Данилевского. Для этого исходная матрица $A$ была преобразована так, что она имеет ненулевые элементы только в первой строке, и элементы на смещённой диагонали равны единице, то есть матрица приведена к форме Фробениуса.

Найдены собственные числа матрицы с помощью WolframAlpha. Полученные решения полностью совпадают.