МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Вычисление значений и корней полиномов

ЛАБОРАТОРНАЯ РАБОТА

студента 4 курса 431 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Серебрякова Алексея Владимировича

Научный руководитель
доцент, к. п. н. ____________ А. С. Гераськин
подпись, дата

Саратов 2022

Определение [править | править код]

Для данной матрицы A, $\chi(\lambda) = \det(A - \lambda E)$, где E — единичная матрица, является многочленом от λ, который называется *характеристическим многочленом* матрицы A (иногда также «вековым уравнением» (англ. *secular equation*)).

Ценность характеристического многочлена в том, что собственные значения матрицы являются его корнями. Действительно, если уравнение $Av = \lambda v$ имеет ненулевое решение, то $(A - \lambda E)v = 0$, значит матрица $A - \lambda E$ вырождена и её определитель $\det(A - \lambda E) = \chi(\lambda)$ равен нулю.

Характеристический полином

определяется для произвольной квадратной матрицы A как[1] $\det(A - \lambda E)$, где E – единичная матрица одинакового с A порядка.

П **Пример.** Для $n = 2$:

$$\det(A - \lambda E) = \begin{vmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{vmatrix} =$$

$$= \lambda^2 - (a_{11} + a_{22})\lambda + (a_{11}a_{22} - a_{12}a_{21});$$

для $n = 3$:

$$\det(A - \lambda E) = \begin{vmatrix} a_{11} - \lambda & a_{12} & a_{13} \\ a_{21} & a_{22} - \lambda & a_{23} \\ a_{31} & a_{32} & a_{33} - \lambda \end{vmatrix} =$$

$$= -\lambda^3 + (a_{11} + a_{22} + a_{33})\lambda^2 -$$

$$-\left\{ \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} + \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} + \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} \right\} \lambda + \det A.$$

```
[alse0722@alse0722-ms7a34 coding_teory]$ /bin/python /home/alse0722/Desktop/univer/coding_teory/fin/n15.py
poly:
[1, -2, 1]

[1.0, 1.0]
[0.0, 0.0]
[alse0722@alse0722-ms7a34 coding_teory]$ 
```

Код программы:

```
import warnings
from sympy.polys.galoistools import gf_monic
from sympy.polys.galoistools import gf_div
from sympy.polys.galoistools import gf_sub_mul
from sympy.polys.galoistools import gf_mul_ground
from sympy.polys.domains import ZZ
import numpy.core.numeric as NX
from numpy.lib.twodim_base import diag
from numpy.linalg import eigvals
from numpy.core import (hstack)
import numpy as np


def roots(p):
    p = np.array(p)
    # find non-zero array entries
    non_zero = NX.nonzero(NX.ravel(p))[0]
```

```
    # Return an empty array if polynomial is all zeros
    if len(non_zero) == 0:
        return NX.array([])

    # find the number of trailing zeros -- this is the number of roots at 0.
    trailing_zeros = len(p) - non_zero[-1] - 1

    # strip leading and trailing zeros
    p = p[int(non_zero[0]):int(non_zero[-1])+1]


    if not issubclass(p.dtype.type, (NX.floating, NX.complexfloating)):
        p = p.astype(float)

    N = len(p)
    if N > 1:
        # наш полином будет являться характеристическим многочленом сопровождающей матрицы
        # Строим сопровождающую матрицу и находим ее собсвтенные значения (корни)
        A = diag(NX.ones((N-2,), p.dtype), -1)
        A[0, :] = -p[1:] / p[0]
        # находим решения характеристического уравнения матрицы
        roots = eigvals(A)
        # которые равны собственным значениям матрицы, то есть найдем корни полинома
    else:
        roots = NX.array([])

    roots = hstack((roots, NX.zeros(trailing_zeros, roots.dtype)))
    return roots

def poly_horner(A, x):
    p = A[-1]
    i = len(A) - 2
    while i >= 0:
        p = p * x + A[i]
        i -= 1
    return p

polynom = [1, -2, 1]

print("poly:")
print(polynom)
print()

rootsPol = roots(polynom)
rootsInt = []

for i in rootsPol:
    rootsInt.append(float(i))

n = len(rootsPol)
print(rootsInt)

values = []
for i in rootsPol:
    values.append(poly_horner(polynom, i))

print(values)
```