In [20]:

```
import seaborn as sns
import scipy as sp
import numpy as np
import cmath
from numpy.polynomial.polynomial import polyval


array_a = np.array([243, -486, 783, -990, 558, -28, -72, 16], dtype=complex)
array_b = np.array([1, 1, 3, 2, -1, -3, -11, -8, -12, -4, -4], dtype=complex)
array_c = np.array([1, complex(0, 1), -1, complex(0, -1), 1], dtype=complex)







def deflate(root, oldPol):
    divider = np.array([1, -root], dtype=complex)
    q, r = np.polydiv(oldPol, divider)
    return np.array(q, dtype=complex)


def laguerre(start, pol):
    z = start
    fun, der, secDer, divisor = 0, 0, 0, 0
    tmp = 100

    while abs(np.polyval(pol, z) - np.polyval(pol, tmp)) > 1e-13:
        tmp = z
        fun = np.polyval(pol, z) * (len(pol) - 1)
        der = np.polyval(np.polyder(pol), z)
        secDer = np.polyval(np.polyder(np.polyder(pol)), z)
        divisor = ((der ** 2 * ((len(pol) - 1) - 1)
                    - (fun * secDer)) * ((len(pol) - 1) - 1)) ** 0.5
        divisor = der - divisor if abs(der - divisor) > abs(der + divisor) else der + divisor
        z = z - (fun / divisor)

    return z


def find_root(polynomial):
    starter = complex(1, 0)
    n = len(polynomial) - 1
    root = np.empty(n, dtype=complex)
    root[0] = laguerre(starter, polynomial)
    poly = polynomial
    i = 0
    while n > 3:
        deflated_poly = deflate(root[i], poly)
        z1 = laguerre(starter, deflated_poly)
        z = laguerre(z1, poly)
        i = i + 1
        root[i] = z
        poly = deflated_poly
        n = len(poly) - 1

    poly = deflate(root[i], poly)
    a = poly[0]
    b = poly[1]
    c = poly[2]

    d = (b ** 2) - (4 * a * c)

    # find two solutions
    sol1 = (-b - cmath.sqrt(d)) / (2 * a)
    sol2 = (-b + cmath.sqrt(d)) / (2 * a)

    i = i + 1
    root[i] = sol1
    i = i + 1
    root[i] = sol2

    for j in range(len(root)):
        print( str(root[j]))


print("Wyniki A")
find_root(array_a)

print(" ")
print("Wyniki B ")
find_root(array_b)

print(" ")
print("Wyniki C: ")
find_root(array_c)
```

```
Wyniki A
(0.6666706057803142+0j)
(0.666668613972958+4.5735602132256395e-06j)
(0.6666646971098434-3.411420399615049e-06j)
(0.3333333333649164+1.8668734899648814e-10j)
(2.6597348691686307e-07-1.414213324847922j)
(-0.33333625104079767-4.171512916209507e-06j)
(-1.2651607213753456e-06+1.4142163340343374j)

Wyniki B
(1.1598346617182338e-08-1.0000000103004474j)
(-3.2354773119288333e-09+0.9999999910394585j)
(-0.49999999999999933-0.8660254037844387j)
(-0.4999999999999997+0.8660254037844384j)
(1.414213562373095-2.909836049025288e-17j)
(-1.1598347101168673e-08-0.9999999896995524j)
(3.2354774319800586e-09+1.000000008960541j)
(-8.818640772162436e-16+1.4142135623730951j)
(-1.414213562373094-3.3306690738754696e-16j)
(-6.661338147750939e-16-1.414213562373094j)

Wyniki C:
(0.9510565162951535+0.30901699437494745j)
(0.5877852522924731-0.8090169943749475j)
(-0.9510565162951534+0.30901699437494734j)
(-0.5877852522924731-0.8090169943749473j)
```

In [ ]: