In [ ]:

```
import math

def f(x, c,n):       # creating the polynomial of degree n
    j = 0
    for i in range(1,n+1):
        j += c[i-1]*x**(n-i)
    return j

def D1f(x, c, n, f):     # first derivative of the polynomial function
    h = 1/1000
    y = (f(x+h,c,n) - f(x-h,c,n))/(2*h)
    return y

def D2f(x, c, n, f):     # second derivative of the polynomial function
    h = 1/1000
    y = (D1f(x+h,c,n,f)-D1f(x-h,c,n,f))/(2*h)
    return y

def Lag(a_o, e, c, n, f):        # laguerre's method
    i = 0 ; a = 0             # i stores the iterations, a is our guess
    while abs(a_o-a) > e and i <= 12:
        a = a_o
        y1 = D1f(a_o, c, n, f)/f(a_o, c, n)       # up upto 2nd term in Taylor expan
        y2 = y1**2 - (D2f(a_o, c, n, f)/f(a_o, c, n))  # up upto 3rd term in Taylor
        if abs(y1 + math.sqrt((n-1)*(n*y2-y1**2))) > abs(y1 - math.sqrt((n-1)*(n*y2-
            k = n/(y1 + math.sqrt((n-1)*(n*y2-y1**2)))
        elif abs(y1 - math.sqrt((n-1)*(n*y2-y1**2))) > abs(y1 + math.sqrt((n-1)*(n*y
            k = n/(y1 - math.sqrt((n-1)*(n*y2-y1**2)))
        else:
            if f(a_o, c, n)==0:
                print('One of the roots obtained:',round(a_o,6))
        a_o -= k       # new trial
        i += 1
    return a_o

def syn_div(a_o, c):      #synthetic division
    if abs(c[0]) != 1:
        for value in c:
                value = value/c[0]          # dividing the coefficients to get coeff
    for k in range(0, len(c)-1):
            c[k+1] = a_o*c[k] + c[k+1]      # separating the values
    return c


def Roots_Laguerre(n ,e, guess, c):            # display function for roots
    for index in range(n, 1, -1):
        guess = Lag(guess, e, c, index, f)     # performing laguerre
        if index > 0:
            c = syn_div(guess,c)        # performing Synthetic division.
        print(round(guess, 4))

print("The roots of the polynomial are:")
Roots_Laguerre(5,0.00001, 1.1, [1, 0, -5, 0, 4])
```

```
The roots of the polynomial are:
1.0
2.0
-1.0
-2.0
```