```python
import numpy as np
from mpmath import *


n = 100  # nombre d'iteracions

Z1 = 0.1 + 0.5*1j  # impedàncies
Z2 = 0.02 + 0.13*1j
Z3 = 0.023 + 0.1*1j
Zp = -10*1j

Y1 = 1 / Z1  # admitàncies
Y2 = 1 / Z2
Y3 = 1 / Z3
Yp = 1 / Zp

P = -1  # dades
Q = -0.1
Va = 1.1

Vb = np. zeros(n, dtype=complex)  # incògnites
Vc = np. zeros(n, dtype=complex)
Vb[0] = 1
Vc[0] = 1

van = 0.5  # dades de la làmpada
lam = 2 * np.sqrt(2) / np.pi
In = np.sqrt(1 - van * van * (2 - lam * lam)) * 1
ang = -np.pi / 2 + np.arctan((van * np. sqrt(lam * lam - van * van))/(1 - van * van)) + np.angle(Vc[0])
Inl = In * cos(ang) + In * sin(ang)*1j

for i in range (1, n):  # iterar per a calcular les incògnites
    Vb[i] = (Va * Y2 + Vc[i - 1] * Y3 + (P - Q*1j) / (np.conj(Vb[i - 1]))) / (Yp + Y2 + Y3)
    Vc[i] = (-Inl + Vb[i] * Y3 + Va * Y1) / (Y1 + Y3)

    ang = -np.pi / 2 + np.arctan((van * np.sqrt(lam * lam - van * van)) / (1 - van * van)) + np.angle(Vc[i])
    Inl = In * cos(ang) + In * sin(ang) * 1j
    I1 = (Va - Vc[i - 1]) / Z1
    I2 = (Va - Vb[i - 1]) / Z2
    I3 = (Vb[i - 1] - Vc[i - 1]) / Z3
    Ipq = np.conj((-P -Q * 1j) / Vb[i - 1])
    Ip = Vb[i - 1] / Zp

    sumatori1 = I2 - (Ip + Ipq + I3)  # balanços d'intensitat
    sumatori2 = I1 + I3 - Inl

I1 = (Va - Vc[n - 1]) / Z1  # càlcul final de corrents
I2 = (Va - Vb[n - 1]) / Z2
I3 = (Vb[n - 1] - Vc[n - 1]) / Z3
Ipq = np.conj((-P -Q * 1j) / Vb[n - 1])
Ip = Vb[n - 1] / Zp

sumatori1 = I2 - (Ip + Ipq + I3)
sumatori2 = I1 + I3 - Inl
print('Balanç 1: ' + str(sumatori1))
print('Balanç 2: ' + str(sumatori2))
```