```
import numpy as np
from mpmath import *


n = 100  # profunditat

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

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))

Vb = np.zeros(n, dtype=complex)  # sèries a calcular
Vc = np.zeros(n, dtype=complex)
R = np.zeros(n, dtype=complex)
X = np.zeros(n, dtype=complex)
F = np.zeros(n, dtype=complex)
L = np.zeros(n, dtype=complex)
Y = np.zeros(n, dtype=complex)
M = np.zeros(n, dtype=complex)
B = np.zeros(n, dtype=complex)
INL = np.zeros(n, dtype=complex)

Vb[0] = Va  # inicialització de les sèries
Vc[0] = (-Va * Y1 - Vb[0] * Y3) / (-Y1 - Y3)
R[0] = 1 / conj(Vb[0])
X[0] = 1 / np.real(Vc[0])
F[0] = np.imag(Vc[0]) * X[0]
B[0] = 1 + F[0] * F[0]
L[0] = np.sqrt(B[0])
Y[0] = 1 / L[0]
M[0] = F[0] * Y[0]
INL[0] = In * 1 * (cos(ang) * Y[0] - sin(ang) * M[0]) + In * 1 *(sin(ang) * Y[0] + cos(ang) * M[0])*1j

sumatori1 = 0
sumatori2 = 0

from Funcions import pade4all


def sumaR(R, Vb, i):  # convolució entre R i Vb
    suma = 0
    for k in range(i):
        suma += R[k] * conj(Vb[i - k])
    return suma


def sumaX(X, Vc, i):  # convolució entre X i Vc real
    suma = 0
    for k in range(i):
        suma += X[k] * np.real(Vc[i - k])
    return suma
```

```
def sumaF(Vc, X, i):  # convolució entre X i Vc imaginari
    suma = 0
    for k in range(i + 1):
        suma += np.imag(Vc[k]) * X[i - k]
    return suma


def sumaY(Y, L, i):  # convolució entre Y i L
    suma = 0
    for k in range(i):
        suma += Y[k] * L[i - k]
    return suma


def sumaM(F, Y, i):  # convolució entre F i Y
    suma = 0
    for k in range(i + 1):
        suma += F[k] * Y[i - k]
    return suma


def sumaB(F, i):  # convolució entre F i la mateixa F
    suma = 0
    for k in range(i + 1):
        suma += F[k] * F[i - k]
    return suma


def sumaL(L, i):  # convolució entre L i la mateixa L
    suma = 0
    for k in range(1, i):
        suma += L[k] * L[i - k]
    return suma


for i in range(1, n):  # càlcul dels coeficients de les sèries
    Vb[i] = ((P - Q*1j) * R[i - 1] - Y3 * (Vb[i - 1] - Vc[i - 1]) - Vb[i - 1] * Yp) / Y2
    Vc[i] = (INL[i - 1] - Vb[i] * Y3) / (-Y1 - Y3)
    R[i] = - sumaR(R, Vb, i) / conj(Vb[0])
    X[i] = - sumaX(X, Vc, i) / np.real(Vc[0])
    F[i] = sumaF(Vc, X, i)
    B[i] = sumaB(F, i)
    L[i] = (B[i] - sumaL(L, i)) / (2 * L[0])
    Y[i] = - sumaY(Y, L, i) / L[0]
    M[i] = sumaM(F, Y, i)
    INL[i] = In * (cos(ang) * Y[i] - sin(ang) * M[i]) + In * (sin(ang) * Y[i] + cos(ang) * M[i]) * 1j

    I1 = (Va - pade4all(i, Vc, 1)) / Z1
    I2 = (Va - pade4all(i, Vb, 1)) / Z2
    I3 = (pade4all(i, Vb, 1) - pade4all(i, Vc, 1)) / Z3
    Ipq = np.conj((-P - Q*1j) / (pade4all(i, Vb, 1)))
    Ip = (pade4all(i, Vb, 1)) / Zp

    sumatori1 = I2 - (Ip + I3 + Ipq)  # balanços d'intensitat
    sumatori2 = I1 + I3 - np.sum(INL)

print('Balanç 1: ' + str(sumatori1))
print('Balanç 2: ' + str(sumatori2))

angvc = np.angle(np.sum(Vc))  # angle de la tensió Vc fonamental

for h in (3, 5, 7, 9):  # càlcul dels harmònics
    Z1 = 0.1 + 0.5 * 1j * h  # noves impedàncies
    Z2 = 0.02 + 0.13 * 1j * h
    Z3 = 0.023 + 0.1 * 1j * h
    Zp = -10 * 1j / h
```

```
Ih = 2 * np.sqrt(2) * 0.5 / (np.pi * h**2)  # intensitat harmònica de la làmpada
angh = h * np.arcsin(0.5 / h) + angvc

Yadm = [[1 / Z2 + 1 / Z3 + 1 / Zp, -1 / Z3], [-1 / Z3, 1 / Z1 + 1 / Z3]]
Yadm = np.array(Yadm)  # matriu d'admitàncies
Ivec = [[0], [-(Ih * np.cos(angh) + Ih * np.sin(angh) * 1j)]]
Ivec = np.array(Ivec)  # vector d'intensitats
Vsol = np.dot(np.linalg.inv(Yadm), Ivec)  # vector de tensions
```