```
#!/usr/bin/env python
# coding: utf-8

# Aqui estão as funções core do problema
# Os gráficos foram ocultados, para diminuir o código


#Import
import scipy.constants as sci
import numpy as np
import matplotlib.pyplot as pl
import math
from mpl_toolkits.mplot3d import Axes3D

from matplotlib import cm
from numpy.linalg import inv
from scipy.linalg import toeplitz


#define constants
E0 = sci.epsilon_0
#U0 = sci.mu_0
Clight = sci.speed_of_light
Freq = 299792458.0 #Hz
Lambda = Clight/Freq
Pi = math.pi
k0 = 2*Pi/Lambda
pl.rcParams['figure.figsize'] = [10, 7]

def voltageColumn(N,V0):
    V = np.zeros((N,1),dtype=np.complex_)
    if(N%2==0):
        temp = int(N/2)
        V[temp] = V0;
        V[temp-1]= V0;
    else:
        temp = int((N-1)/2)
        V[temp] = V0
    return V

def funcZ(x,L,N):
    Delta = L/(N+1)
    if(x < 0):
        x = 0
    return -L/2 + x*Delta


def funcPsi(m,n, L, N,a):
    Delta = L/(N+1)
    if(m == n):
        fst = 1./(2*Pi*Delta)
        snd = np.log((Delta/2 + np.sqrt( (Delta/2)**2 + a**2  ) ) /a )
        third = 1j*k0/4*Pi
        return fst*snd - third
    else:
        fst = np.sqrt( (funcZ(m,L,N) - funcZ(n,L,N))**2 + a**2 )
        snd = np.exp(-1j*k0*fst)
        third = 4*Pi
        return snd/(third*fst)



def funcAmn(m,n,L,N,a):
    Delta = L/(N+1)
    return (Delta**2)*funcPsi(m,n,L,N,a)
```

```
def funcPhi(m,n,L,N,a):
    fst = funcPsi(m-1/2 , n-1/2,L,N,a)
    snd = funcPsi(m+1/2 , n-1/2,L,N,a)
    thd = funcPsi(m-1/2 , n+1/2,L,N,a)
    qrt = funcPsi(m+1/2 , n+1/2,L,N,a)
    return (fst - snd - thd + qrt)


def impedanceMtz(L,N,a):
    Zmn = np.empty([N,N], dtype=np.complex_)
    ZmnAux = np.zeros((1,N), dtype=np.complex_)
    for i in range(0,N):
        for j in range (0,N):
            Zmn[i][j] = (k0**2)*funcAmn(i+1,j+1,L,N,a) - funcPhi(i+1,j+1,L,N,a)
    for i in range(0,N):
        ZmnAux[0][i] = (k0**2)*funcAmn(1,i+1,L,N,a) - funcPhi(1,i+1,L,N,a)
    return Zmn, ZmnAux


def entryCurrent(L,N,a,V0):
    Zmtz, Zaux = impedanceMtz(L,N,a)
    Zaux = toeplitz(Zaux)
    ZauxInv = inv(Zaux)
    ZInv = inv(Zmtz)
    Vin = voltageColumn(N,V0)
    Imnaux = np.dot(ZauxInv,Vin)
    Imn = np.dot(ZInv,Vin)
    return Imn[int((N-1)/2)]


def entryImpedance(L,N,a,V0):
    Zmtz, Zaux = impedanceMtz(L,N,a)
    Zaux = toeplitz(Zaux)
    ZauxInv = inv(Zaux)
    ZInv = inv(Zmtz)
    Vin = voltageColumn(N,V0)
    Imnaux = np.dot(ZauxInv,Vin)
    Imn = np.dot(ZInv,Vin)
    return 1./Imn[int((N-1)/2)]

def main():
    #Exemplos de uso
    L = Lambda/2
    N = 19
    a = Lambda*(10**(-4))
    V0 = -1j*2*Pi*Freq*E0
    print (entryCurrent(L,N,a,V0))
    print (entryImpedance(L,N,a,V0))

    L = Lambda
    print (entryCurrent(L,N,a,V0))
    print (entryImpedance(L,N,a,V0))


if __name__ == '__main__':
    main()
```