Relatório do Projeto Computacional de Eletromagnetismo

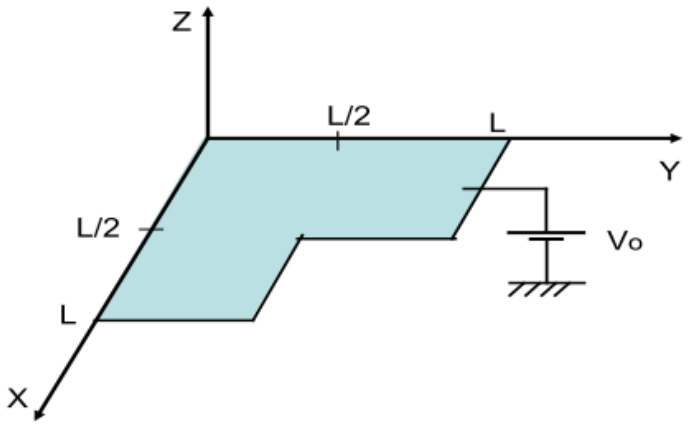
Amanda Moraes

Abril 2021

1 Problema

Uma placa condutora de lado maior L é mantida a um potencial V_0, como mostrado abaixo. Assuma que o meio seja espaço livre. Aplique o método dos momentos para determinar a distribuição superficial de carga na placa. Para tanto divida cada lado L em N segmentos (N par).

Figura 1: Placa condutora



1. Expandindo a distribuição de carga em funções de base tipo pulso, determine as expressões para os elementos das matrizes de impedância e de tensão do método dos momentos.
2. Para o caso em que L = 10 cm, e V0 = 1 V, resolva o sistema linear para um valor de N específico (você escolhe, mas par). Determine as amplitudes dos pulsos, e obtenha uma aproximação para a distribuição de carga superficial na placa. Plote o resultado.
3. Resolva o problema e plote a distribuição superficial de carga para diferentes valores de N. Comente os resultados.
4. Determine a carga total na placa. Varie N e observe a convergência.

2 Solução

Sabe-se que o potencial na superfície condutora é constante e portanto:

$$V(\overrightarrow{r}) = \frac{1}{4\pi\varepsilon_0}\iint_{S'}\frac{\rho_s(\overrightarrow{r'})}{|\overrightarrow{r}-\overrightarrow{r'}|}ds' = V_0 \qquad \forall\overrightarrow{r} \in placa \qquad (I)$$

A distribuição de carga ρ_s na placa não é conhecida e pode ser determinada por uma aproximação:

Primeiro a placa condutora deve ser discretizada em pequenas regiões quadradas com lados iguais a $\delta = L/N$. Assim, a área total da superfície em L ilustrada na Figura 1 abrigará $3N^2/4$ elementos de área.

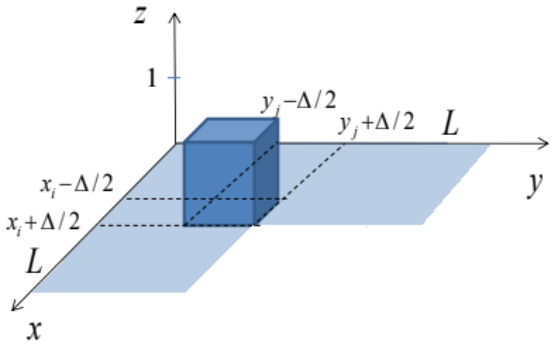
Em seguida, a densidade superficial de carga pode ser aproximada como o somatório de funções pulso $P(x, y)$ multiplicadas por suas amplitudes a_n (incógnitas do sistema) nos elementos quadrados obtidos com a discretização da superfície condutora.

Para cada elemento de superfície n, a função pulso é definida por:

$$P_n(x,y) = \begin{cases} 1 \text{ para } x_i - \delta/2 < x < x_i + \delta/2, \quad y_j - \delta/2 < y < y_j + \delta/2 \\ 0 \text{ para pontos fora do elemento de área } n \end{cases}$$

onde $n \in \{1, 2, ...\frac{3N^2}{4}\}$ e (x_i, y_j) é o ponto central do quadrado n em questão.

Figura 2: Placa condutora no plano xy e função pulso para um quadrado de lado δ



Aproximação da distribuição de carga ao longo da placa condutora:

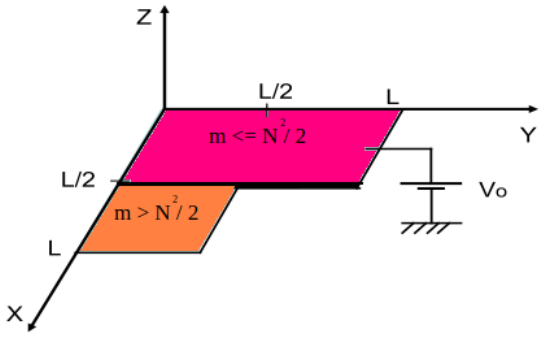
$$\rho_s(x,y) \approx \sum_{n=1}^{3N^2/4} a_n P_n(x,y) \qquad (II)$$

com a_n sendo a amplitude do pulso do n-ésimo quadrado.

Para determinar a distribuição superficial de carga aproximada na placa com potencial V_0, será considerada a condição de contorno nos pontos centrais $\overrightarrow{r_m}$ dos quadrados que compõem a superfície condutora. Esses pontos podem ser definidos por:

$$\overrightarrow{r_m} = (x_p, y_q) = ((p-1/2)\delta, (q-1/2)\delta) \begin{cases} p = (m \mod \frac{N}{2}) + 1 \\ q = \frac{N}{2} + (m - \frac{N^2}{2})/\frac{N}{2} + 1 \quad \text{se } m > \frac{N^2}{2} \\ \text{(região laranja da Figura 3)} \\ \\ p = (m \mod N) + 1 \\ q = \frac{m}{N} + 1 \quad \text{se } m \leq \frac{N^2}{2} \\ \text{(região rosa da Figura 3)} \end{cases} \quad \text{com } m = 1, 2, ..., \frac{3N^2}{4}$$

Figura 3: Regiões consideradas para obter os índices das coordenadas dos pontos centrais de cada quadrado m



Finalmente, de (I) e (II) temos que:

$$V_0 \approx \frac{1}{4\pi\varepsilon_0} \iint_{S'} \frac{\sum_{n=1}^{(3N^2/4)} a_n P_n(x', y')}{|\overrightarrow{r_m} - \overrightarrow{r'}|} \, ds' \Rightarrow$$

$$V_0 \approx \sum_{n=1}^{(3N^2/4)} a_n \frac{1}{4\pi\varepsilon_0} \int_{x_i-\delta/2}^{x_i+\delta/2} \int_{y_j-\delta/2}^{y_j+\delta/2} \frac{1}{|\overrightarrow{r_m} - \overrightarrow{r'}|} \, dx' \, dy' \qquad \text{com } m = 1, 2, ..., \frac{3N^2}{4} \qquad (III)$$

A última expressão pode ser escrita em forma matricial como:

$$\begin{bmatrix} \\ Z_{mn} \\ \\ \end{bmatrix} \begin{bmatrix} a_1 \\ . \\ . \\ . \\ a_m \end{bmatrix} = \begin{bmatrix} \\ V_m \\ \\ \end{bmatrix}$$

$$m \in 1, 2, ..., 3N^2/4$$

$$n \in 1, 2, ..., 3N^2/4$$

Sendo $[Z]$ a matriz de impedância de dimensões $m \times n$, $[a]$ a matriz de amplitudes dos pulsos de dimensões $m \times 1$ e $[V]$ a matriz de tensão de dimensões $m \times 1$.

2.1 Questão 1

De (III) temos que os elementos da matriz de impedância e da matriz de tensão correspondem a:

$$\begin{cases} V_m = V_0 \\ Z_{mn} = \frac{1}{4\pi\varepsilon_0} \int_{x_i-\delta/2}^{x_i+\delta/2} \int_{y_j-\delta/2}^{y_j+\delta/2} \frac{1}{|\overrightarrow{r_m} - \overrightarrow{r'}|} \, dx' \, dy' \end{cases}$$

No código desenvolvido em Python, a matriz de tensão é inicializada com o seguinte trecho:

```
# Matriz de tensao (mesmo valor Vo para cada elemento quadrado, que
    sao 3(N**2)/4 elementos)
matTensao = np.ones(shape=((3*(N**2)//4),1)) * Vo
```

Para determinar Z_{mn} aproximado quando $m \neq n$ podemos considerar que toda a carga do quadrado n está concentrada em seu ponto central:

$$Z_{mn} = \frac{1}{4\pi\varepsilon_0} \int_{x_i-\delta/2}^{x_i+\delta/2} \int_{y_j-\delta/2}^{y_j+\delta/2} \frac{1}{\sqrt{(x_p - x')^2 + (y_q - y')^2}} \, dx' \, dy' \Rightarrow$$

$$Z_{mn} = \frac{1}{4\pi\varepsilon_0} \int_{x_i-\delta/2}^{x_i+\delta/2} \int_{y_j-\delta/2}^{y_j+\delta/2} \frac{1}{\sqrt{(x_p - x_i)^2 + (y_q - y_j)^2}} \, dx' \, dy' \Rightarrow$$

$$Z_{mn} \approx \frac{1}{4\pi\varepsilon_0} \frac{\delta^2}{\sqrt{(x_p - x_i)^2 + (y_q - y_j)^2}}$$

Quando $m = n$, o ponto (x_p, y_q) é o mesmo ponto representado por (x_i, y_j) na expressão de Z_{mn}:

$$Z_{mn} = \frac{1}{4\pi\varepsilon_0} \int_{x_i-\delta/2}^{x_i+\delta/2} \int_{y_j-\delta/2}^{y_j+\delta/2} \frac{1}{\sqrt{(x_p - x')^2 + (y_q - y')^2}} \, dx' \, dy' \Rightarrow$$

$$Z_{mn} = \frac{1}{4\pi\varepsilon_0} \int_{x_i-\delta/2}^{x_i+\delta/2} \int_{y_j-\delta/2}^{y_j+\delta/2} \frac{1}{\sqrt{(x_i - x')^2 + (y_j - y')^2}} \, dx' \, dy' \Rightarrow$$

$$\dots$$

$$Z_{mn} = \frac{\delta}{\pi\varepsilon_0} ln(1 + \sqrt{2})$$

Logo, Z_{mn} pode ser aproximado para:

$$Z_{mn} = \begin{cases} \frac{1}{4\pi\varepsilon_0} \frac{\delta^2}{\sqrt{(x_p - x_i)^2 + (y_q - y_j)^2}} & \text{se } m \neq n \\ \frac{\delta}{\pi\varepsilon_0} ln(1 + \sqrt{2}) & \text{se } m = n \end{cases}$$

No código desenvolvido, a matriz de impedância é inicializada com o seguinte trecho:

```
# a matriz de impedancia estabelece uma relacao de cada quadradinho
    com os outros, por isso tem dimensao 3(N**2)/4 x 3(N**2)/4
matImped = np.ones(shape=((3*(N**2)//4),(3*(N**2)//4)))

for m in range(matImped.shape[0]): # de 0 a (3*(N**2)//4
  if (m > (N*(N/2))):
```

```
    p = m%(N/2) + 1
    q = (N/2) + (m - (N*(N/2)))//(N/2) + 1
  else:
    p = m%N + 1
    q = m//N + 1

  # determina o ponto medio do primeiro quadrado em questao
  xp = p * delta - delta/2
  yq = q * delta - delta/2

  for n in range(matImped.shape[1]): # de 0 a (3*(N**2)//4
    if (n > (N*(N/2))):
      xiindex = n%(N/2) + 1
      yjindex = (N/2) + (n - (N*(N/2)))//(N/2) + 1
    else:
      xiindex = n%N + 1
      yjindex = n//N + 1

    # determina o ponto medio do segundo quadrado em questao
    xi = xiindex * delta - delta/2
    yj = yjindex * delta - delta/2

    # valor de Zmn para elementos fora da diagonal da matriz de
    impedancia
    if m != n:
      distPQtoIJ = math.sqrt((xp - xi)**2 + (yq - yj)**2)
      matImped[m][n] = (1/(4 * math.pi * epsilon)) * ((delta**2)/
    distPQtoIJ)
    # valor de Zmn para elementos da diagonal da matriz de
    impedancia
    else:
      matImped[m][n] = (delta/(math.pi * epsilon)) * np.log(1 +
    math.sqrt(2))
```

2.2 Questão 2

A partir desse ponto, todos os valores das matrizes $[Z]$ e $[V]$ são conhecidos e podemos obter os valores de $[a]$ por inversão de matrizes:

$$[Z][a] = [V] \Rightarrow [a] = [Z]^{-1}[V]$$

Ao calcular a inversa da matriz de impedância e fazer seu produto com a matriz de tensão, temos a amplitude do pulso para cada pequeno elemento de área da placa. Esse resultado representa uma aproximação da distribuição de carga superficial ao longo do objeto condutor ilustrado na Figura 1.

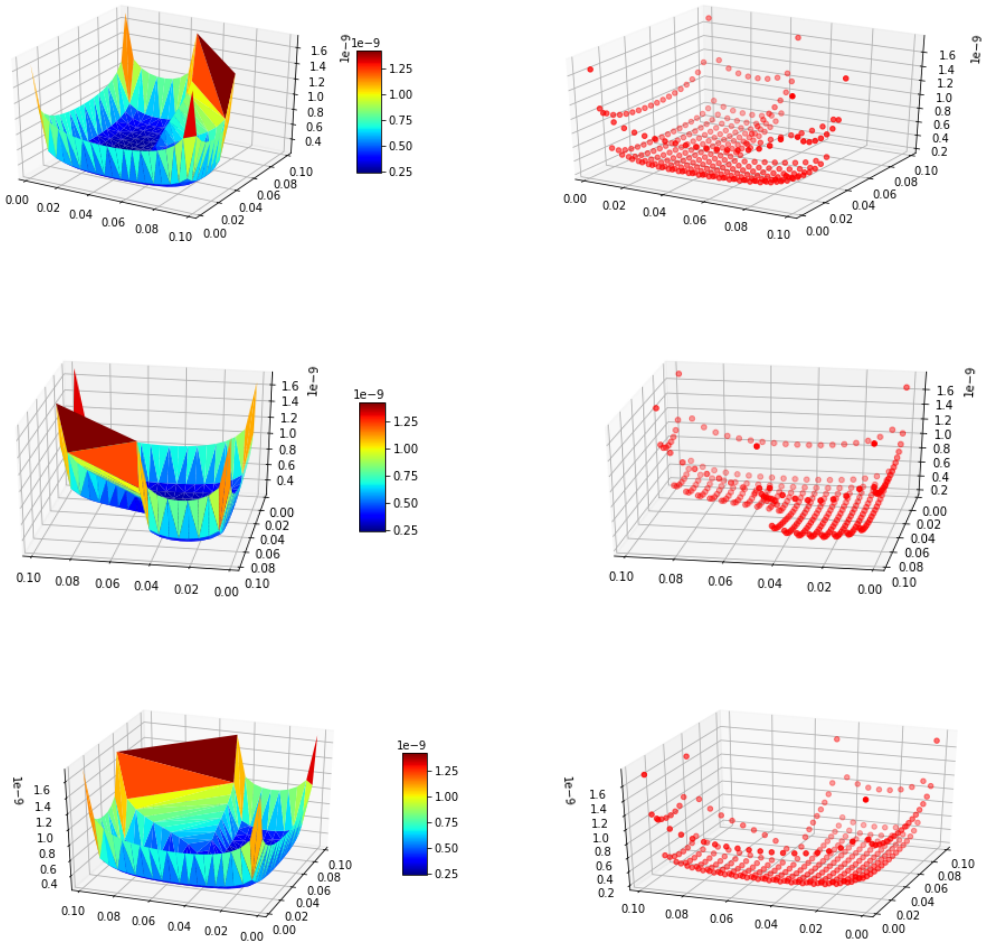
Em Python:

```
matImpedInv = np.linalg.inv(matImped) # inversa da matriz de
    impedancia
A = np.dot(matImpedInv, matTensao) # amplitudes
```

Para $L = 10cm$, $V_0 = 1V$ e $N = 20$:

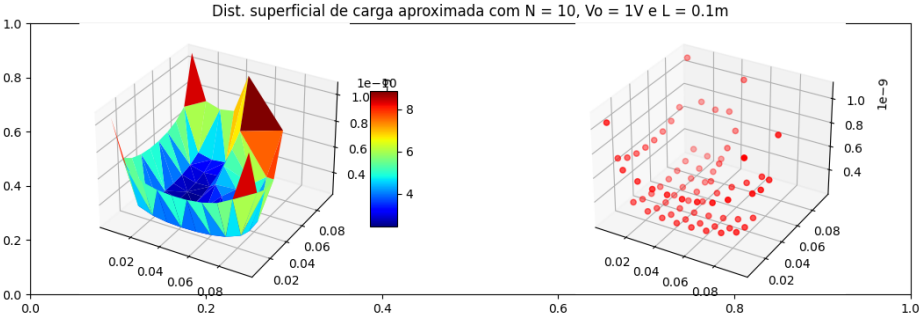
```
# Parametros
L = 0.1 # em metros
N = 20 # numero natural par
Vo = 1 # potencial em V
delta = L/N
```

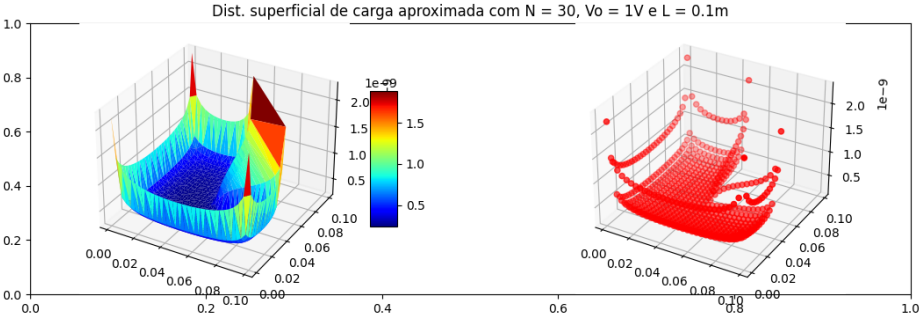
Resultado de diferentes ângulos:

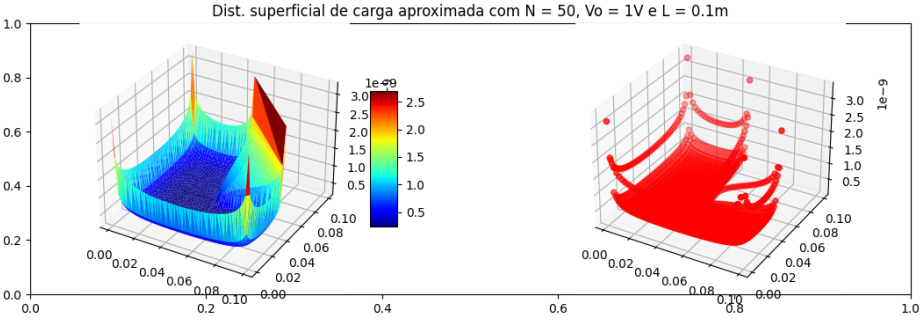


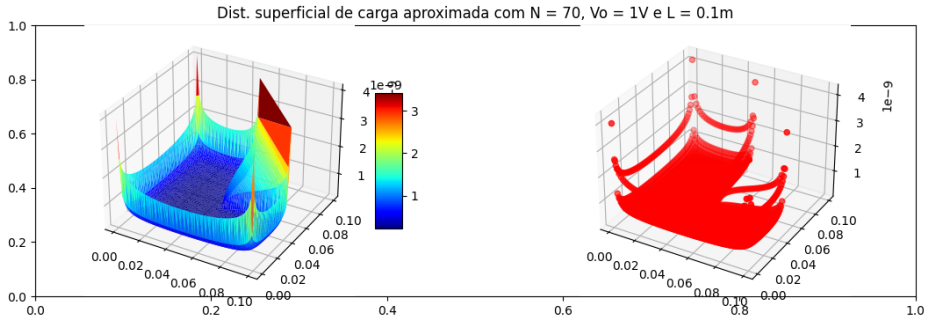
2.3 Questão 3

Ao calcular e plotar a distribuição superficial de carga aproximada para diferentes valores de N (10, 30, 50 e 70) percebe-se que, embora o gráfico apresente maior grau de precisão (e mais uniformidade na superfície), o custo computacional e temporal do método dos momentos aumenta consideravelmente para os valores mais altos (há mais elementos de área a serem considerados nas operações).


Dist. superficial de carga aproximada com N = 10, Vo = 1V e L = 0.1m
1e−10
1e−9


Dist. superficial de carga aproximada com N = 30, Vo = 1V e L = 0.1m
1e−9
1e−9


Dist. superficial de carga aproximada com N = 50, Vo = 1V e L = 0.1m
1e−9
1e−9



2.4 Questão 4

O cálculo da carga total foi aproximado como o somatório da distribuição superficial em cada elemento quadrado de área:

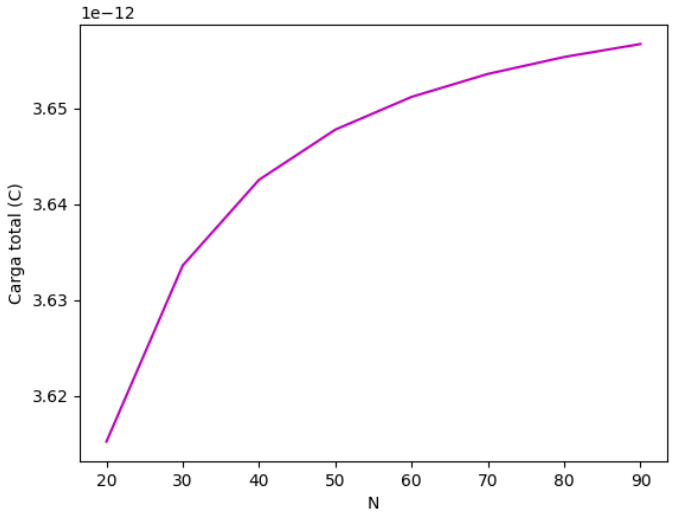
$$\int \rho_S ds \approx \sum_{n=1}^{3N^2/4} \rho_n$$

Trecho de código correspondente:

```
def getCargaTotal(self):
    # soma todos os elementos da matriz de amplitudes multiplicada
    por delta^2 ((L/N)^2)
    return np.sum(self.A * (self.L/self.N)**2)
```

A carga total aproximada na placa condutora para diferentes valores de N, como apresentado na Figura 4, resulta num valor cada vez mais preciso à medida que N aumenta. Pelo gráfico obtido, a carga total na superfície converge para $3,66 \times 10^{-12} C$.

Figura 4: Gráfico N x Carga total da placa (aproximada)



3 Código completo

O código completo e a estrutura de diretórios utilizada estão disponíveis para visualização e download no repositório ProjEletromag do meu GitHub (junto com os comandos necessários para sua execução):

`https://github.com/amandascm/ProjEletromag`

Os arquivos principais do trabalho computacional também foram inseridos abaixo:

Classe desenvolvida para representar a placa condutora e suas características:

```
import numpy as np
import math
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
from matplotlib import cm

class Placa:
    def __init__(self, L = 0.1, N = 20, Vo = 1):
        # Parametros
        self.L = L # em metros
        self.N = N # numero natural par
        self.Vo = Vo # potencial em V

        # Matriz de tensao (um valor para cada centro de elemento quadrado, que s o 3(N**2)/4 elementos)
        self.matTensao = np.ones(shape=((3*(N**2)//4),1)) * Vo

        # Coordenadas dos centros de cada quadrado obtido com discretizacao
        self.xCoords = []
        self.yCoords = []

        # Define a matriz de impedancia e as coordenadas dos centros de cada elemento de area
        self.discretCalcMatImped()

        # Define a matriz de amplitudes A
        self.calcAmp()
        self.zCoords = list(self.A.reshape((self.A.shape[0])))

    def plotDistCarga(self):
        # set up a figure twice as wide as it is tall
        fig = plt.figure(figsize=plt.figaspect(0.25))
        plt.title(f"Dist. superficial de carga aproximada com N = {self.N}, Vo = {self.Vo}V e L = {self.L}m")
        #===============
        #  First subplot (surface)
        ax = fig.add_subplot(1, 2, 1, projection='3d')
        surf = ax.plot_trisurf(self.xCoords, self.yCoords, self.zCoords, cmap=cm.jet, linewidth=0.1)
        fig.colorbar(surf, shrink=0.5, aspect=5)
        #===============
        # Second subplot (points)
        ax = fig.add_subplot(1, 2, 2, projection='3d')
        ax.scatter(self.xCoords,self.yCoords,self.zCoords, c="red")
        plt.show()
        fig.savefig(f"../imgs/distCarga/grafN{self.N}")

    def getCargaTotal(self):
        return np.sum(self.A * (self.L/self.N)**2)

```

```
47      def calcAmp(self):
48          matImpedInv = np.linalg.inv(self.matImped) # inversa da
        matriz de impedancia
49          self.A = np.dot(matImpedInv, self.matTensao) # amplitudes
50
51      def discretCalcMatImped(self):
52          N = self.N
53          delta = self.L/N
54          epsilon = 8.85*1e-12
55
56          # a matriz de impedancia estabelece uma relacao de cada
        quadradinho com todos os outros, por isso tem dimensao 3(N**2)
        /4 x 3(N**2)/4
57          matImped = np.ones(shape=((3*(N**2)//4),(3*(N**2)//4)))
58          for m in range(matImped.shape[0]):
59              if (m > (N*(N/2))):
60                  p = m%(N/2) + 1
61                  q = (N/2) + (m - (N*(N/2)))//(N/2) + 1
62              else:
63                  p = m%N + 1
64                  q = m//N + 1
65
66              xp = p * delta - delta/2
67              yq = q * delta - delta/2
68
69              self.xCoords += [xp]
70              self.yCoords += [yq]
71
72              for n in range(matImped.shape[1]):
73                  if (n > (N*(N/2))):
74                      xiindex = n%(N/2) + 1
75                      yjindex = (N/2) + (n - (N*(N/2)))//(N/2) + 1
76                  else:
77                      xiindex = n%N + 1
78                      yjindex = n//N + 1
79
80                  xi = xiindex * delta - delta/2
81                  yj = yjindex * delta - delta/2
82
83                  if m != n:
84                      distPQtoIJ = math.sqrt((xp - xi)**2 + (yq - yj)
        **2)
85                      matImped[m][n] = (1/(4 * math.pi * epsilon)) *
        ((delta**2)/distPQtoIJ)
86                  else:
87                      matImped[m][n] = (delta/(math.pi * epsilon)) *
        np.log(1 + math.sqrt(2))
88          self.matImped = matImped
```

Classe principal responsável por instanciar e invocar métodos da classe placa e gerar as respostas das questões propostas:

```
1  import placa
2  import matplotlib.pyplot as plt
3
4  def Q2():
5      print("----------Q2-----------")
6      print("L = 0.1m\nN = 20\nVo = 1\n")
7      p = placa.Placa()
8      p.plotDistCarga()
9
10 def Q3():
11     print("----------Q3-----------")
```

```
12      listN = [10,30,50,70]
13      for n in listN:
14          print(f"------\nN = {n}")
15          p = placa.Placa(N=n)
16          p.plotDistCarga()
17
18  def Q4():
19      print("----------Q4-----------")
20      listQ = []
21      listN = range(20,100,10)
22      for n in listN:
23          print(f"------\nN = {n}")
24          p = placa.Placa(N=n)
25          listQ += [p.getCargaTotal()]
26      fig = plt.figure(1)
27      plt.plot(listN, listQ, 'm')
28      plt.xlabel('N')
29      plt.ylabel('Carga total (C)')
30      plt.title('Carga total aproximada obtida com diferentes valores
         de N')
31      plt.show()
32      fig.savefig(f"../imgs/carga/grafCarga")
33
34  def main():
35      Q2()
36      Q3()
37      Q4()
38
39  if __name__ == "__main__":
40      main()
```