Lista 4 - Potencial Elétrico

Q1

Usando a trajetória $x^2 = 8y$ vamos encontrar o trbaalho necessário para mover a carga $Q = -20\mu C$ da origem até o ponto (4,2,0) a partir do campo elétrico dado no enunciado com a ideia de que trabalho é o produto da força pela distância

$$W = \int_C \vec{F} \cdot d\vec{r}$$

Sabendo que $\vec{F} = q\vec{E}$ podemos subistituir e ficamos com

$$W = q \int \int \vec{E} \cdot d\vec{r}$$

$$W = q \iint ((2x + 8y)\ \vec{ax} + 8x\ \vec{ay}) \cdot (dx\ \vec{ax} + dy\ \vec{ay})$$

Como temos que fazer pelo caminho $x^2 = 8y$

$$W = q \iint ((2x + x^2)\ \vec{ax} + 8\sqrt{8y}\ \vec{ay}) \cdot (dx\ \vec{ax} + dy\ \vec{ay})$$

In [361…
```
from math import *
import sympy as sp
q = -20e-6

x, y = sp.symbols('x y')

W = q*(sp.integrate(2*x+x**2,(x,0,4)) + sp.integrate(8*sp.sqrt(8*y),(y,0,2)))
W
print("W = {:.2e} J".format(W))
```

```
W = -1.60e-3 J
```

$$W = -1.6mJ$$

Q4

Sabendo que a densidade de fluxo é dado por

$$D = \epsilon_0 E$$

e

$$E = -\nabla V$$

podemos calcular a densidade em $D = (2, \frac{\pi}{2}, 0)$

In [362…
```
r,O,phi = sp.symbols("r theta phi")
ar,aO,aphi = sp.symbols('a_r a_theta a_phi')
V = lambda r,O,phi: (10/r**2)*sp.sin(O)*sp.cos(phi)
V(r,O,phi)
```

Out[362…

$$\frac{10 \sin(\theta) \cos(\phi)}{r^2}$$

In [363…

```
Er = V(r,O,phi).diff(r) *ar
EO = (1/r)*V(r,O,phi).diff(O)*aO
Ephi = (1/(r*sp.sin(O))*(V(r,O,phi).diff(phi)))*aphi

E = -(Er+EO+Ephi)
E
```

Out[363…

$$\frac{10 a_\phi \sin(\phi)}{r^3} + \frac{20 a_r \sin(\theta) \cos(\phi)}{r^3} - \frac{10 a_\theta \cos(\phi) \cos(\theta)}{r^3}$$

In [364…

```
e0 = 8.85e-12
D = e0*E.subs(r,2).subs(O,sp.pi/2).subs(phi,0)
D
```

Out[364…

$$2.2125 \cdot 10^{-11} a_r$$

Sabendo que

$$W = -qV_{AB}$$

e adotando uma carga $Q = 10\mu C$

In [365…

```
q = 10e-6
grau = lambda g: (g*sp.pi)/180 #convertendo para radianos

W = -q*(V(1,grau(30),grau(120))-V(4,grau(90),grau(60)))
print('W = {:.4e} J'.format(W))
```

```
W = 2.8125e-5 J
```

Q8

Sabendo que

$$V(r) = \frac{Q}{4\pi\epsilon_0 r}$$

In [366…

```
Q = 500e-12
ra = 5
rb = 15

V = lambda r: Q/(4*sp.pi*e0*r)
Vab = V(ra)- V(rb)
Vab.evalf(3)
```

Out[366…

$$0.599$$

$$V_A - V_B = 0.6V$$

Q11

Sabendo que $dQ = \rho_l \, dl$ então como a imagem nos mostra um cilindro planificado, podemos pensar em uma espécie de capacitor cilindrico é usar

$$V = -\int_{r_A}^{r_B} \frac{\rho_l \, dr}{2\pi\epsilon_0 r}$$

In [367…

```
import numpy as np
ra = np.array([0,-4,4])
rb = np.array([0,-5,12])
rho = 20e-12
modulo = lambda r: np.sqrt(sum((r)**2))
rA = modulo(ra)
rB = modulo(rb)

f = rho/(2*sp.pi*e0*r)
V = - sp.integrate(f,(r,rB,rA))
V.evalf(3)
```

Out[367… 0.299

$$V_{AB} = 0.3V$$

Q10

A energia armazenada no cilindro é dada por

$$U = \frac{\epsilon_0}{2} \int_V E^2 \, dV$$

onde $dV = r \, dr \, d\theta \, dz$

In [368…

```
r,z = sp.symbols('r z')
r1 = 0.01
r2 = 0.05
l = 0.5

E = 1e5/r

U = (e0/2) * sp.integrate(sp.integrate(sp.integrate(r*E**2,(r,r1,r2)),(O,0,2*sp.pi))
U.evalf()
```

Out[368… 0.22373677575345

Energia armazenada é $U = 0.22J$

Q5

Partindo do conhecimento de que o potencial V pode ser escrito usando o momento de dipolo

$$V = \frac{Q \ d \ \cos\theta}{4\pi\epsilon_0 \ r^2} = \frac{\vec{p} \cdot a_r}{4\pi\epsilon_0 \ r^2}$$

onde a_r é o vetor unitário que aponta para o sentido da distância até o ponto determinado para a pedição do potencial, na origem

$$V = \frac{\vec{p} \cdot \vec{r}}{4\pi\epsilon_0 \ r^3}$$

In [369…

```
ra = np.array([0,0,2]) #criando o vetor a partir da origem
rb = np.array([0,0,-3]) #criando o vetor a partir da origem
az = np.array([0,0,1])
```

```
pa = -5e-9*az
pb = 9e-9*az

V = lambda r,p: np.dot(p,r)/(4*np.pi*e0*(modulo(r)**3))  ##função modulo, foi criada

v =  V(ra,pa) + V(rb,pb)

v
```

Out[369…

```
-20.23156056252907
```

por tanto

$$V = -20.23V$$

Q9

Sabendo que

$$\vec{E} = -\nabla V$$

e Dado v somente em ϕ, $V = -\frac{60\phi}{\pi}$

In [370…

```
V = -60*phi/sp.pi
E = lambda phi: (1/r)*V.diff(phi) ## considerando apenas a parte modular
E(phi)
```

Out[370…

$$-\frac{60}{\pi r}$$

o volume infinitezimal é

$$dV = r\,dr\,d\phi\,dz$$

e que a energia armazenada é dada por

$$U = \frac{\epsilon_0}{2}\int_V E^2\,dV$$

In [371…

```
U = (e0/2) * (sp.integrate(sp.integrate(sp.integrate(r*E(phi)**2,(r,0.1,0.6)),(phi,0
U.evalf()
```

Out[371…

$$1.51423876846818 \cdot 10^{-9}$$

Logo a energia armazenada é

$$U = 1.514nJ$$

Q6

Dado que a capacitância em um capacitor de placas paralelas e planas

$$C = \epsilon_0\frac{A}{d}$$

onde A é a área do plano do capacitor e d é a distância entre as placas.

Usando a lei de Gauss agora já evetuada as devidas considerações temos que

$$E = \frac{q}{\epsilon_0 A}$$

onde A é a área da gaussiana

Uma vez encontrado o campo podemos a energia armazenada

$$U = \frac{\epsilon_0}{2}\int_V E^2 \, dV$$

em uma situação de equilibrio eletrostático o campo eletrico em placas paralelas pode ser $E = \frac{V}{d}$ sabendo que a tensão é $V = \frac{Q}{C}$ podemos relacionar e ficamos com

$$E = \frac{Q}{C \cdot d}$$

Logo

$$U = \frac{\epsilon_0}{2}\left(\frac{Q}{Cd}\right)^2 \int_V dV$$

$$U = \frac{\epsilon_0 \, Q^2 A \, d}{2 \, C^2 \, d^2}$$

$$U = \frac{V^2}{2}\frac{\epsilon_0 \, A}{d}$$

$$U = \frac{CV^2}{2}$$

Q3

Sabendo que o potencial Elétrico numa linha de carga infinita é dado por

$$V_L = -\frac{\rho_L}{2\pi\epsilon_0} ln(r)$$

e na carga pontual

$$V_Q = \frac{Q}{4\pi\epsilon_0}\frac{1}{r}$$

em cada ponto tempos

$$V_{ponto} = V_L + V_Q$$

In [372…

```
#entradas
A = np.array([5,0,1])
O = np.array([0,0,0])
Vo = 0

#pontos
La = np.array([A[0],1,1]) #ponto na linha de carga
Lo = np.array([O[0],1,1])

Q = np.array([-3,4,0])

#constantes
```

```
k = 1/(4*sp.pi*e0)
rho = 2e-9
q = 5e-9

#
rQA = A - Q ## distância perp de Q até A
rLA = A - La ## distância perp da linha até A

rQO = O - Q ## distância perp de Q até B
rLO = O - Lo ## distância perp de Q até B


Vl = lambda r: - (rho/(2*sp.pi*e0)) *sp.ln(modulo(r))

Vq = lambda r: (k*q)/modulo(r)

Va = - (Vl(rLO) + Vq(rQO) -  Vl(rLA) - Vq(rQA) + Vo )
Va.evalf()
```

Out[372... 8.46893050223656

Feito o algoritmo a cima, podemos somente mudar os pontos que as projeções algebricas vão se acertar automaticamente logo para letra b

In [373...

```
C = np.array([-2,5,3])
B = np.array([1,2,1])
Vb = 100

Lc = np.array([C[0],1,1])
Lb = np.array([B[0],1,1])


rQC = C - Q
rLC = C - Lc
rQB = B - Q
rLB = B - Lb


Vc =  (Vl(rLC) + Vq(rQC) -  Vl(rLB) - Vq(rQB) + Vb )

Vc.evalf()
```

Out[373... 49.8707133274791

Já na letra C temos

In [374...

```
Vbc = Vc - Vb
Vbc.evalf()
```

Out[374... -50.1292866725209

Considerando todas as unidades de medidas em volts