Lista 3 - RCO

September 25, 2022

1 Questão 1

Nota: o algoritmo aqui mostrado funciona independentemente do número de lâminas ser par ou ímpar.

1.1 Bibliotecas utilizadas:

[ ]:
```
import numpy as np
import math
import matplotlib.pyplot as plt

# Para mostrar apenas 3 casas decimais
np.set_printoptions(precision=3)
```

1.2 Constantes e dados de entrada:

[ ]:
```
# Direções: informar aqui a direção de cada lâmina (em graus), de baixo para␣
↪cima em relação ao laminado:
direc_deg = [45,0,0,45]

# Conversão para radianos
direc = np.radians(direc_deg)
n = len(direc)

#Vetor de carregamentos (elementos não-nulos estão em unidades de N/mm)
# Informar aqui os esforços, obedecendo a convenção:
#carreg = [[Nx],[Ny],[Nxy],[Mx],[My],[Mxy]]

carreg = [[1000],[200],[0],[0],[0],[0]]

#Espessura de cada lâmina (esp): informar
esp = 3E-3 #m

#Espessura do laminado
h = n*esp #m



# Dados do material: informar aqui:
```

```
E11 = 19.76E9 #Pa
E22 = 1.97E9 #Pa
nu12 = 0.35
G12 = 0.7E9 #Pa

#Relação entre coeficientes de Poisson
nu21 = (E22*nu12)/E11
```

```
[0.785 0.    0.    0.785]
```

1.3 Matriz de rigidez reduzida transformada no sistema global de coordenadas:

[ ]:
```
Q11 = E11/(1-nu12*nu21)
Q22 = E22/(1-nu12*nu21)
Q66 = G12
Q12 = (nu12*E22)/(1-nu12*nu21)
Q21 = Q12

Q = np.array([[Q11, Q12, 0],[Q21, Q22, 0], [0, 0, Q66]])

cos = np.cos(direc)
sin = np.sin(direc)

# T = np.zeros((1,4))
# T_inv = np.zeros((1,4))
# Q_dash = np.zeros((4,1))

# Inicialização dos vetores
T = [[0 for _ in range(1)] for _ in range(n)]
T_inv = [[0 for _ in range(1)] for _ in range(n)]
Q_dash = [[0 for _ in range(1)] for _ in range(n)]

# Cálculo da matriz Q_dash para cada uma das lâminas
for i in range(n):
    T[i] = np.array([[cos[i]**2, sin[i]**2, 2*sin[i]*cos[i]],[sin[i]**2,␣
 ↪cos[i]**2, -2*sin[i]*cos[i]],[-sin[i]*cos[i], sin[i]*cos[i],␣
 ↪cos[i]**2-sin[i]**2]])
    T_inv[i] = np.linalg.inv(T[i])
    Q_dash[i] = T_inv[i]@Q@T[i]
```

1.4 Matriz "ABBD":

[ ]:
```
# Matriz A: rigidez à tração e compressão
# Inicialização
A_local = [[0 for _ in range(1)] for _ in range(n)]
A_global = 0
```

```
for i in range(n):
    A_local[i] = Q_dash[i]*(((((n/2)-(i+1))/n)*-h) - (((n/2 - i)/n)*-h))
    #A_local[i] = Q_dash[i]*(h/n) # retorna os mesmos resultados, significa que␣
 ↪está ok
    A_global = A_global + A_local[i]
#print(A_global)

# Matriz B: acoplamento entre rigidez no plano e rigidez à flexão
B_local = [[0 for _ in range(1)] for _ in range(n)]
B_global = 0
for i in range(n):
    B_local[i] = 0.5*Q_dash[i]*(((((n/2)-(i+1))/n)*-h)**2 - (((n/2 - i)/
 ↪n)*-h)**2)
    B_global = B_global + B_local[i]
#print(B_global)

# Matriz D: rigidez à flexão ou torção
D_local = [[0 for _ in range(1)] for _ in range(n)]
D_global = 0
for i in range(n):
    D_local[i] = (1/3)*Q_dash[i]*(((((n/2)-(i+1))/n)*-h)**3 - (((n/2 - i)/
 ↪n)*-h)**3)
    D_global = D_global + D_local[i]
#print(D_global)

# Combinando as matrizes em uma só
linha1 = np.vstack((A_global,B_global))
linha2 = np.vstack((B_global,D_global))
ABBD = np.hstack((linha1,linha2))
print("Matriz ABBD:")
print(ABBD)
```

```
Matriz ABBD:
[[1.572e+08 3.718e+07 5.403e+07 1.455e-11 0.000e+00 0.000e+00]
 [3.718e+07 4.916e+07 5.403e+07 0.000e+00 0.000e+00 0.000e+00]
 [2.701e+07 2.701e+07 6.601e+07 0.000e+00 0.000e+00 0.000e+00]
 [1.455e-11 0.000e+00 0.000e+00 1.141e+03 7.054e+02 1.135e+03]
 [0.000e+00 0.000e+00 0.000e+00 7.054e+02 8.169e+02 1.135e+03]
 [0.000e+00 0.000e+00 0.000e+00 5.673e+02 5.673e+02 1.311e+03]]
```

1.5 Deformações no plano médio e curvatura em relação ao sistema global:

[ ]:
```
# Inicialização do vetor
def_curv = [[0],[0],[0],[0],[0],[0]]

def_curv = np.linalg.inv(ABBD)@carreg
```

```python
epsilon_0_global = np.vstack((def_curv[0],def_curv[1],def_curv[2]))
K_global = np.vstack((def_curv[3],def_curv[4],def_curv[5]))

# Tensões e deformações em cada lâmina (coordenadas z referenciadas no plano␣
↪médio de cada lâmina)
sigma_global = [[0 for _ in range(1)] for _ in range(n)]
sigma_local = [[0 for _ in range(1)] for _ in range(n)]
z = [[0 for _ in range(1)] for _ in range(n)]
epsilon_global = [[0 for _ in range(1)] for _ in range(n)]
epsilon_local = [[0 for _ in range(1)] for _ in range(n)]
y = [[0 for _ in range(1)] for _ in range(n)]

for i in range(n):
    z[i] = 0.5*(((((n/2)-(i+1))/n)*-h) + (((n/2 - i)/n)*-h))
    sigma_global[i] = (Q_dash[i]@(epsilon_0_global + (z[i]*K_global)))*10**(-3)
    print("==============================")
    print("Resultados lâmina %d:" %(i+1))
    print("Tensão na lâmina no sistema global de coordenadas (kPa): ")
    print(sigma_global[i])
    sigma_local[i] = T[i]@sigma_global[i]
    print("Tensão na lâmina no sistema local de coordenadas (kPa):")
    print(sigma_local[i])
    epsilon_global[i] = epsilon_0_global + z[i]*K_global
    print("Deformação no plano médio da lâmina no sistema global de coordenadas:
↪")
    print(epsilon_global[i])
    epsilon_local[i] = T[i]@epsilon_global[i]
    print("Deformação no plano médio da lâmina no sistema local de coordenadas:
↪")
    print(epsilon_local[i])
```

```
==============================
Resultados lâmina 1:
Tensão na lâmina no sistema global de coordenadas (kPa):
[[23.949]
 [21.482]
 [ 3.008]]
Tensão na lâmina no sistema local de coordenadas (kPa):
[[25.724]
 [19.707]
 [-1.234]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 7.013e-06]
 [ 3.488e-06]
```

```
 [-4.298e-06]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[ 9.527e-07]
 [ 9.548e-06]
 [-1.762e-06]]
==============================
Resultados lâmina 2:
Tensão na lâmina no sistema global de coordenadas (kPa):
[[142.718]
 [ 11.851]
 [ -3.008]]
Tensão na lâmina no sistema local de coordenadas (kPa):
[[142.718]
 [ 11.851]
 [ -3.008]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 7.013e-06]
 [ 3.488e-06]
 [-4.298e-06]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[ 7.013e-06]
 [ 3.488e-06]
 [-4.298e-06]]
==============================
Resultados lâmina 3:
Tensão na lâmina no sistema global de coordenadas (kPa):
[[142.718]
 [ 11.851]
 [ -3.008]]
Tensão na lâmina no sistema local de coordenadas (kPa):
[[142.718]
 [ 11.851]
 [ -3.008]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 7.013e-06]
 [ 3.488e-06]
 [-4.298e-06]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[ 7.013e-06]
 [ 3.488e-06]
 [-4.298e-06]]
==============================
Resultados lâmina 4:
Tensão na lâmina no sistema global de coordenadas (kPa):
[[23.949]
 [21.482]
 [ 3.008]]
Tensão na lâmina no sistema local de coordenadas (kPa):
```

```
[[25.724]
 [19.707]
 [-1.234]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 7.013e-06]
 [ 3.488e-06]
 [-4.298e-06]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[ 9.527e-07]
 [ 9.548e-06]
 [-1.762e-06]]
```

1.6 Rascunhos

[ ]:
```
### Teste se o resultado era o mesmo pra ABBD

# def_curv_2 = [[0],[0],[0],[0],[0],[0]]

# A_est = np.linalg.inv(A_global)
# B_est = -1*np.linalg.inv(A_global)@B_global
# C_est = B_global@np.linalg.inv(A_global)
# D_est = D_global-B_global@np.linalg.inv(A_global)@B_global

# A_ap = A_est-B_est@np.linalg.inv(D_est)@C_est
# B_ap = B_est@np.linalg.inv(D_est)
# C_ap = B_ap
# D_ap = np.linalg.inv(D_est)

# print(B_ap)

# linha1_2 = np.vstack((A_ap,B_ap))
# linha2_2 = np.vstack((C_ap,D_ap))
# ABBD_2 = np.hstack((linha1_2,linha2_2))

# def_curv_2 = ABBD_2@carreg
# print(def_curv)
# print(def_curv_2)




# y[i] = epsilon_local[i][1]

# plt.figure()
# plt.plot(z, y)
```