Lista_3_RCO_questao1

October 1, 2022

Autor: Norton Martin Reichert Trennepohl

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)
```

[ ]:
```
# Direções: informar aqui a direção de cada lâmina (em graus), de CIMA para␣
↪BAIXO 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*1000],[200*1000],[0],[0],[0],[0]] #N/m

#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
```

1.2 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)

# 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)]
Reuter = [[1,0,0],[0,1,0],[0,0,2]] # Matriz de Reuter

# 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@Reuter@T[i]@np.linalg.inv(Reuter)
#   print(Q_dash[i])
```

1.3 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:
[[1.593e+08 3.508e+07 2.701e+07 1.455e-11 0.000e+00 0.000e+00]
 [3.508e+07 5.126e+07 2.701e+07 0.000e+00 0.000e+00 0.000e+00]
 [2.701e+07 2.701e+07 3.510e+07 0.000e+00 0.000e+00 0.000e+00]
 [1.455e-11 0.000e+00 0.000e+00 1.185e+03 6.613e+02 5.673e+02]
 [0.000e+00 0.000e+00 0.000e+00 6.613e+02 8.610e+02 5.673e+02]
 [0.000e+00 0.000e+00 0.000e+00 5.673e+02 5.673e+02 6.616e+02]]
```

Como era esperado, a matriz [B] é nula, já que o laminado é simétrico.

1.4 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

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]))
```

```
print(def_curv)
```

```
[[ 6.874e-03]
 [ 3.340e-03]
 [-7.861e-03]
 [-1.621e-16]
 [ 7.566e-17]
 [ 7.412e-17]]
```

Percebe-se que os valores de curvatura são muito baixos (podem ser considerados como nulos), o que já era esperado pois não é realizado momento sobre a estrutura. Isso indica que os resultados são condizentes.

1.5 Tensões e deformações em cada lâmina:

[ ]:
```
# 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**(-6)
    print("==============================")
    print("Resultados lâmina %d:" %(i+1))
    print("Tensão na lâmina no sistema global de coordenadas (MPa): ")
    print(sigma_global[i])
    sigma_local[i] = T[i]@sigma_global[i]
    print("Tensão na lâmina no sistema local de coordenadas (MPa):")
    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 (MPa):
[[26.822]
 [21.874]
 [ 5.502]]
Tensão na lâmina no sistema local de coordenadas (MPa):
[[29.85 ]
 [18.845]
 [-2.474]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 0.007]
 [ 0.003]
 [-0.008]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[-0.003]
 [ 0.013]
 [-0.002]]
==============================
Resultados lâmina 2:
Tensão na lâmina no sistema global de coordenadas (MPa):
[[139.845]
 [ 11.46 ]
 [ -5.502]]
Tensão na lâmina no sistema local de coordenadas (MPa):
[[139.845]
 [ 11.46 ]
 [ -5.502]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 0.007]
 [ 0.003]
 [-0.008]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[ 0.007]
 [ 0.003]
 [-0.008]]
==============================
Resultados lâmina 3:
Tensão na lâmina no sistema global de coordenadas (MPa):
[[139.845]
 [ 11.46 ]
 [ -5.502]]
Tensão na lâmina no sistema local de coordenadas (MPa):
[[139.845]
 [ 11.46 ]
 [ -5.502]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 0.007]
 [ 0.003]
 [-0.008]]
```

```
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[ 0.007]
 [ 0.003]
 [-0.008]]
==============================
Resultados lâmina 4:
Tensão na lâmina no sistema global de coordenadas (MPa):
[[26.822]
 [21.874]
 [ 5.502]]
Tensão na lâmina no sistema local de coordenadas (MPa):
[[29.85 ]
 [18.845]
 [-2.474]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 0.007]
 [ 0.003]
 [-0.008]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[-0.003]
 [ 0.013]
 [-0.002]]
```

Lista_3_RCO_questao2

October 1, 2022

1 Questão 2

1.1 Constantes e dados de entrada:

[ ]:
```
# Direções: informar aqui a direção de cada lâmina (em graus), de CIMA para␣
↪BAIXO em relação ao laminado:
direc_deg = [0,45,90,0,45,90]

# 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 = [[100*1000],[0],[0],[0],[0],[0]] #N/m

#Espessura de cada lâmina (esp): informar
esp = 3E-4 #m

#Espessura do laminado
h = n*esp #m



# Dados do material: informar aqui:
E11 = 155000E6 #Pa
E22 = 12100E6 #Pa
nu12 = 0.35
G12 = 4400E6 #Pa

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

1.2 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)

# 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)]
Q_26 = [[0 for _ in range(1)] for _ in range(n)]
Reuter = [[1,0,0],[0,1,0],[0,0,2]]

# 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@Reuter@T[i]@np.linalg.inv(Reuter)
#    print(Q_dash[i])
```

1.3 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)
    #print(((((n/2)-(i+1))/n)*-h)) #- (((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:
[[ 1.305e+08  2.908e+07  2.164e+07 -2.597e+04  2.274e-13  1.184e-14]
 [ 2.908e+07  1.305e+08  2.164e+07  2.274e-13  2.597e+04  1.976e-12]
 [ 2.164e+07  2.164e+07  2.930e+07  1.184e-14  1.976e-12  1.137e-13]
 [-2.597e+04  2.274e-13  1.184e-14  3.651e+01  6.569e+00  4.545e+00]
 [ 2.274e-13  2.597e+04  1.976e-12  6.569e+00  3.651e+01  4.545e+00]
 [ 1.184e-14  1.976e-12  1.137e-13  4.545e+00  4.545e+00  6.629e+00]]
```

Obs: os elementos de expoente negativo muito alto (10^{-13}, 10^{-14} etc.) podem ser considerados nulos.

1.4 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

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]))

print("Matriz deformação/curvatura:")
print(def_curv)
```

```
Matriz deformação/curvatura:
[[ 1.078e-03]
```

```
 [-1.259e-04]
 [-7.031e-04]
 [ 8.372e-01]
 [ 1.135e-02]
 [-5.817e-01]]
```

1.5 Tensões e deformações em cada lâmina:

```
[ ]: # 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))
         np.set_printoptions(precision=3)
         sigma_global[i] = (Q_dash[i]@(epsilon_0_global + (z[i]*K_global)))*10**(-6)
         print("==============================")
         print("Resultados lâmina %d:" %(i+1))
         print("Tensão na lâmina no sistema global de coordenadas (MPa): ")
         print(sigma_global[i])
         sigma_local[i] = T[i]@sigma_global[i]
         print("Tensão na lâmina no sistema local de coordenadas (MPa):")
         print(sigma_local[i])

         np.set_printoptions(precision=6)
         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 (MPa):
[[69.85 ]
 [ 0.282]
```

```
 [-1.174]]
Tensão na lâmina no sistema local de coordenadas (MPa):
[[69.85 ]
 [ 0.282]
 [-1.174]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 0.00045 ]
 [-0.000134]
 [-0.000267]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[ 0.00045 ]
 [-0.000134]
 [-0.000267]]
==============================
Resultados lâmina 2:
Tensão na lâmina no sistema global de coordenadas (MPa):
[[13.011]
 [ 5.688]
 [ 2.896]]
Tensão na lâmina no sistema local de coordenadas (MPa):
[[12.246]
 [ 6.454]
 [-3.662]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 0.000701]
 [-0.000131]
 [-0.000441]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[-0.000156]
 [ 0.000726]
 [-0.000416]]
==============================
Resultados lâmina 3:
Tensão na lâmina no sistema global de coordenadas (MPa):
[[ 11.089]
 [-15.898]
 [ -2.71 ]]
Tensão na lâmina no sistema local de coordenadas (MPa):
[[-15.898]
 [ 11.089]
 [  2.71 ]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 0.000952]
 [-0.000128]
 [-0.000616]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[-0.000128]
 [ 0.000952]
```

```
 [ 0.000616]]
==============================
Resultados lâmina 4:
Tensão na lâmina no sistema global de coordenadas (MPa):
[[187.807]
 [  3.629]
 [ -3.477]]
Tensão na lâmina no sistema local de coordenadas (MPa):
[[187.807]
 [  3.629]
 [ -3.477]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 0.001203]
 [-0.000124]
 [-0.00079 ]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[ 0.001203]
 [-0.000124]
 [-0.00079 ]]
==============================
Resultados lâmina 5:
Tensão na lâmina no sistema global de coordenadas (MPa):
[[31.24 ]
 [17.376]
 [ 9.478]]
Tensão na lâmina no sistema local de coordenadas (MPa):
[[33.786]
 [14.83 ]
 [-6.932]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
[[ 0.001455]
 [-0.000121]
 [-0.000965]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[-0.000298]
 [ 0.001632]
 [-0.000788]]
==============================
Resultados lâmina 6:
Tensão na lâmina no sistema global de coordenadas (MPa):
[[ 20.337]
 [-11.077]
 [ -5.013]]
Tensão na lâmina no sistema local de coordenadas (MPa):
[[-11.077]
 [ 20.337]
 [  5.013]]
Deformação no plano médio da lâmina no sistema global de coordenadas:
```

```
[[ 0.001706]
 [-0.000117]
 [-0.001139]]
Deformação no plano médio da lâmina no sistema local de coordenadas:
[[-0.000117]
 [ 0.001706]
 [ 0.001139]]
```