Homeworks for SM-I course

Important:

1). Please download the latest grcodes, images, and related chapters before working on the homework.

2). Both pdf files and the source codes must be submitted, or the work will not be marked.

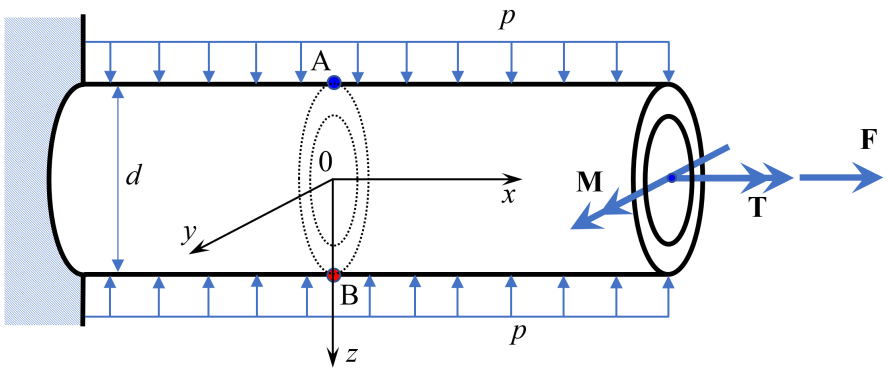
Homework 8:

Question: Understanding failure criteria

Consider a hollow cylindrical shaft with an outer diameter of $d_O = 100$ mm, and an inner diameter of $d_I = 80$ mm. It is loaded by a torque $\mathbf{T} = 28$ kNm, a bending moment $\mathbf{M} = 18$ kNm, an axial force $\mathbf{F} = 100$ kN, and a pressure $p = 20$ MPa on its outer surface. The directions of these loads are shown in the image below. The shaft is made of steel treated as linear elastic material before yielding. Its Young's modulus is $E = 180GPa$, and Poisson's ratio is $\nu = 0.32$. Assume the safety factor is $S_f = 2.8$, and the material yield stress $\sigma_Y = 720MPa$.

Determine whether or not the shaft will yield at point A, using

1. The maximum principle stress criterion.
2. The maximum shear stress criterion.
3. The von Mises yield criterion.
4. Repeat items 1)-3) for point B.



A hollow cylindrical shaft subjected to multiple loads.

In [ ]:
```
# Place curse in this cell, and press Ctrl+Enter to import dependences.
import sys                          # for accessing the computer system
sys.path.append('../grbin/')  # Change to the directory in your system

from commonImports import *       # Import dependences from '../grbin/'
import grcodes as gr                 # Import the module of the author
#importlib.reload(gr)             # When grcodes is modified, reload it

from continuum_mechanics import vector
init_printing(use_unicode=True)      # For latex-like quality printing
np.set_printoptions(precision=4,suppress=True,
       formatter={'float': '{:0.4e}'.format})   # Digits in print-outs
```

In [ ]:
```
def principalS(S):
    '''Compute the principal stresses and their directions.
    inputs:
       S: given stress tensor, numpy array
    return:
       principal stresses (eigenValues), their direction cosines
       (eigenVectors) ranked by its values. Right-hand-rule is enforced
    '''
    eigenValues, eigenVectors = lg.eig(S)

    #Sort in order
    idx = eigenValues.argsort()[::-1]
    eigenValues = eigenValues[idx]
    eigenVectors = eigenVectors[:,idx]
    print('Pricipal stress (Eigenvalues):\n',eigenValues,'\n')

    # make the first element in the first vector positive (optional):
    #eigenVectors[0,:] = eigenVectors[0,:]/np.sign(eigenVectors[0,0])

    # Determine the sign for given eigenVector-1 and eigenVector-3
    eigenVectors[:,2] = np.cross(eigenVectors[:,0], eigenVectors[:,1])

    angle = np.arccos(eigenVectors[0,0])*180/np.pi       # in degree
    print(f'Principal stress directions:\n{eigenVectors}\n')
    print(f"Possible angles (n1,x)={angle}◦ or {180-angle}◦")

    return eigenValues, eigenVectors
```

Point A

1.

In [ ]:
```
# External forces
Torque = 28e3 # Nm
Moment = 18e3 # Nm
p = 20e6      # Pa, pressure on the surface
SF = 2.8      # safety factor

# Material property
E = 180e9     # Pa, Young's modulus, not used in this example
v = 0.32      # Poisson's ratio
sY= 720e6     # Pa, Yield stress

# Dimensions of the cylindrical shaft
do = 0.1      # m, outer diameter
dI = 0.08     # m, inner diameter

# Compute structure/member properties
I = (np.pi * (do**4 - dI**4))/64.0  # 2nd moment of circular area
J = (np.pi * (do**4 - dI**4))/32.0  # 2nd polar moment of circular area

# Compute the stresses
s =-SF*Moment*((do/2)-(dI/2))/I   # stress by the moment at point A
t = SF*Torque*((do/2)-(dI/2))/J   # stress by the Torque

sxx=s; sxy=t; sxz=0; syy=0; syz=0; szz=SF*p

Sij = np.array([[sxx, sxy, sxz],[sxy, syy, syz],[sxz, syz, szz]])
print(f'Stress state:\n{Sij}\n')

eigens, eigenVectors = principalS(Sij)      # principal stresses
```

```
Stress state:
[[-1.7391e+08 1.3526e+08 0.0000e+00]
 [1.3526e+08 0.0000e+00 0.0000e+00]
 [0.0000e+00 0.0000e+00 5.6000e+07]]

Pricipal stress (Eigenvalues):
 [7.3845e+07 5.6000e+07 -2.4775e+08]

Principal stress directions:
[[-4.7919e-01 0.0000e+00 -8.7771e-01]
 [-8.7771e-01 0.0000e+00 4.7919e-01]
 [0.0000e+00 1.0000e+00 0.0000e+00]]

Possible angles (n1,x)=118.6323868639462◦ or 61.367613136053805◦
```

In [ ]:
```
# maximum principal strain criterion
se1 = eigens[0]-v*eigens[1]-v*eigens[2]  # principal strains E ε1
se2 = eigens[1]-v*eigens[0]-v*eigens[2]  # principal strains E ε2
se3 = eigens[2]-v*eigens[0]-v*eigens[1]  # principal strains E ε3
e_max =max(abs(np.array([se1, se2, se3])))
print(f'The maximum principal strain is {e_max:0.5e}')
print(f'The material yield stress needs to be > {e_max:0.5e}')
print(f'The yield stress of the material = {sY:0.5e}')
print(f'Is yield? {e_max>sY}')
print(f'Actual safety factor = {SF*sY/e_max}')
```

```
The maximum principal strain is 2.89302e+08
The material yield stress needs to be > 2.89302e+08
The yield stress of the material = 7.20000e+08
Is yield? False
Actual safety factor = 6.968499875359853
```

2.

In [ ]:
```python
# Maximum shear stress (Tresca) criterion
Tau_max =0.5*(eigens[0]-eigens[2])
print(f'The maximum shear stress is {Tau_max:0.5e}')
print(f'The material yield stress needs to be > {(2*Tau_max):0.5e}')
print(f'The yield stress of the material = {sY:0.5e}')
print(f'Is yield? {(2*Tau_max)>sY}')
print(f'Actual safety factor = {SF*sY/(2*Tau_max)}')
```

```
The maximum shear stress is 1.60798e+08
The material yield stress needs to be > 3.21597e+08
The yield stress of the material = 7.20000e+08
Is yield? False
Actual safety factor = 6.268719881338143
```

3.

In [ ]:
```python
# von Mises criterion
svm = gr.von_Mises(eigens)
print(f'The von Mises stress is {svm:0.5e}')
print(f'The material yield stress needs to be > {svm:0.5e}')
print(f'The yield stress of the material = {sY:0.5e}')
print(f'Is yield? {svm>sY}')
print(f'Actual safety factor = {SF*sY/svm}')
```

```
The von Mises stress is 3.13056e+08
The material yield stress needs to be > 3.13056e+08
The yield stress of the material = 7.20000e+08
Is yield? False
Actual safety factor = 6.43974760140724
```

Point B

1.

In [ ]:
```
# External forces
Torque = 28e3 # Nm
Moment = 18e3 # Nm
p = 20e6      # Pa, pressure on the surface
SF = 2.8      # safety factor

# Material property
E = 180e9     # Pa, Young's modulus, not used in this example
v = 0.32      # Poisson's ratio
sY= 720e6     # Pa, Yield stress

# Dimensions of the cylindrical shaft
do = 0.1      # m, outer diameter
dI = 0.08     # m, inner diameter

# Compute structure/member properties
I = (np.pi * (do**4 - dI**4))/64.0  # 2nd moment of circular area
J = (np.pi * (do**4 - dI**4))/32.0  # 2nd polar moment of circular area

# Compute the stresses
s =-SF*Moment*((do/2)-(dI/2))/I   # stress by the moment at point B
t = SF*Torque*((do/2)-(dI/2))/J   # stress by the Torque

sxx=s; sxy=t; sxz=0; syy=0; syz=0; szz=SF*p

Sij = np.array([[sxx, sxy, sxz],[sxy, syy, syz],[sxz, syz, szz]])
print(f'Stress state:\n{Sij}\n')

eigens, eigenVectors = principalS(Sij)      # principal stresses
```

```
Stress state:
[[-1.7391e+08 1.3526e+08 0.0000e+00]
 [1.3526e+08 0.0000e+00 0.0000e+00]
 [0.0000e+00 0.0000e+00 5.6000e+07]]

Pricipal stress (Eigenvalues):
 [7.3845e+07 5.6000e+07 -2.4775e+08]

Principal stress directions:
[[-4.7919e-01 0.0000e+00 -8.7771e-01]
 [-8.7771e-01 0.0000e+00 4.7919e-01]
 [0.0000e+00 1.0000e+00 0.0000e+00]]

Possible angles (n1,x)=118.6323868639462◦ or 61.367613136053805◦
```

In [ ]:
```
# maximum principal strain criterion
se1 = eigens[0]-v*eigens[1]-v*eigens[2]  # principal strains E ε1
se2 = eigens[1]-v*eigens[0]-v*eigens[2]  # principal strains E ε2
se3 = eigens[2]-v*eigens[0]-v*eigens[1]  # principal strains E ε3
e_max =max(abs(np.array([se1, se2, se3])))
print(f'The maximum principal strain is {e_max:0.5e}')
print(f'The material yield stress needs to be > {e_max:0.5e}')
print(f'The yield stress of the material = {sY:0.5e}')
print(f'Is yield? {e_max>sY}')
print(f'Actual safety factor = {SF*sY/e_max}')
```

```
The maximum principal strain is 2.89302e+08
The material yield stress needs to be > 2.89302e+08
The yield stress of the material = 7.20000e+08
Is yield? False
Actual safety factor = 6.968499875359853
```

2.

In [ ]:
```
# Maximum shear stress (Tresca) criterion
Tau_max =0.5*(eigens[0]-eigens[2])
print(f'The maximum shear stress is {Tau_max:0.5e}')
print(f'The material yield stress needs to be > {(2*Tau_max):0.5e}')
print(f'The yield stress of the material = {sY:0.5e}')
print(f'Is yield? {(2*Tau_max)>sY}')
print(f'Actual safety factor = {SF*sY/(2*Tau_max)}')
```

```
The maximum shear stress is 1.60798e+08
The material yield stress needs to be > 3.21597e+08
The yield stress of the material = 7.20000e+08
Is yield? False
Actual safety factor = 6.268719881338143
```

3.

In [ ]:
```
# von Mises criterion
svm = gr.von_Mises(eigens)
print(f'The von Mises stress is {svm:0.5e}')
print(f'The material yield stress needs to be > {svm:0.5e}')
print(f'The yield stress of the material = {sY:0.5e}')
print(f'Is yield? {svm>sY}')
print(f'Actual safety factor = {SF*sY/svm}')
```

```
The von Mises stress is 3.13056e+08
The material yield stress needs to be > 3.13056e+08
The yield stress of the material = 7.20000e+08
Is yield? False
Actual safety factor = 6.43974760140724
```