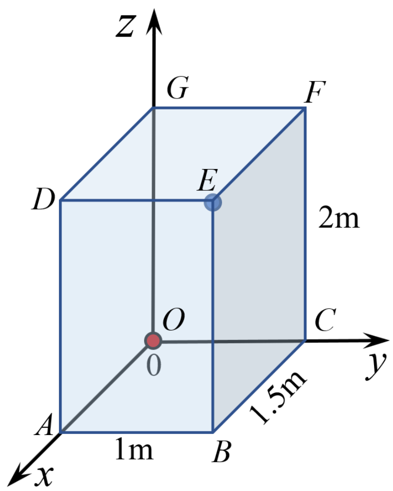
Homeworks for SM-I course

Homework 3: Understanding displacements, strains and coordinate transformation

Consider a 3D solid brick shown



The displacements are given in the following formulas.

$$
\begin{aligned}
u &= c_1 + c_2 xyz \\
v &= c_3 + c_4 xyz \\
w &= c_5 + c_6 xyz
\end{aligned} \tag{1}
$$

where $c_1, c_2, \ldots, c_6$ are constants. Through a measurement, the displacements at point E are found as $(0.004, 0.002, -0.004)$ (m), and point F are found as $(-0.004, -0.002, 0.004)$ (m).

1. Determine the functions for all the displacement components.

2. Compute the gradient of the displacement vector functions.

3. Compute the strain functions in the solid, and the values of the strains at pint E.

4. Compute the normal strains at point E along the $\overrightarrow{OE}$ direction, and that along $\overrightarrow{CE}$ direction.

5. Compute the shear strains at point E between $\overrightarrow{EG}$ and $\overrightarrow{EB}$.

6. Compute the principal strains, and strain invariants.

7. Rotate the coordinates by 30° about y-axis, and find the displacements at point E in the new coordinates system.

8. Rotate the coordinates by 30° about y-axis, and find the strains at point E in the new coordinates system.

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)  # Digits in print-outs
```

1.

In [ ]:
```
x, y, z = symbols("x, y, z")             # define symbolic coordinates
c1, c2, c3, c4, c5, c6 = symbols("c_1, c_2, c_3, c_4, c_5, c_6")
u = c1+c2*x*y*z
v = c3+c4*x*y*z
w = c5+c6*x*y*z
xE = {x:3/2, y:1, z:2}                      # Coordinates at point E
xF = {x:0, y:1, z:2}                        # Coordinates at point F
dE = [0.004, 0.002, -0.004]   # × 1e-3                 # displacment at point E
dF = [-0.004, -0.002, 0.004]  # x 1e-3                 # displacment at point F
sln_cs1=sp.solve([u.subs(xE)-dE[0],v.subs(xE)-dE[1],w.subs(xE)-dE[2]],
                [c1, c3, c5])
u = u.subs(sln_cs1)
v = v.subs(sln_cs1)
w = w.subs(sln_cs1)
sln_cs2=sp.solve([u.subs(xF)-dF[0],v.subs(xF)-dF[1],w.subs(xF)-dF[2]],
                [c2, c4, c6])
```

In [ ]:
```
U = Matrix([u.subs(sln_cs2), v.subs(sln_cs2), w.subs(sln_cs2)])
gr.printM(U.T, 'The displacement functions are found as:')
```

```
The displacement functions are found as:
```

Out[ ]:
$$\begin{bmatrix} 0.002667xyz - 0.004 & 0.001333xyz - 0.002 & -0.002667xyz + 0.004 \end{bmatrix}$$

2.

In [ ]:
```
np.set_printoptions(precision=4,suppress=True)

# the gradient of the displacement vector functions
U_g = vector.grad_vec(U)
printM(U_g, 'Displacement gradient', n_dgt=4)
```

```
Displacement gradient
```

Out[ ]:
$$\begin{bmatrix} 0.002667yz & 0.001333yz & -0.002667yz \\ 0.002667xz & 0.001333xz & -0.002667xz \\ 0.002667xy & 0.001333xy & -0.002667xy \end{bmatrix}$$

3.

In [ ]:
```
strains = 0.5*(U_g +U_g.T)
printM(strains,'Strain tensor (small) functions', n_dgt=4)
```

```
Strain tensor (small) functions
```

Out[ ]:
$$\begin{bmatrix} 0.002667yz & 0.001333xz + 0.0006667yz & 0.001333xy - 0.001333yz \\ 0.001333xz + 0.0006667yz & 0.001333xz & 0.0006667xy - 0.001333xz \\ 0.001333xy - 0.001333yz & 0.0006667xy - 0.001333xz & -0.002667xy \end{bmatrix}$$

In [ ]:
```
Ev = strains.subs(xE)                    # values of the strains
printM(Ev, 'Strain tensor values:', n_dgt=4)
```

```
Strain tensor values:
```

Out[ ]:
$$\begin{bmatrix} 0.005333 & 0.005333 & -0.0006667 \\ 0.005333 & 0.004 & -0.003 \\ -0.0006667 & -0.003 & -0.004 \end{bmatrix}$$

4.

In [ ]:
```python
Ev = np.array(Ev)

# normal strain at point E along OE direction:
N_OE = np.array([1.5/np.sqrt((1.5**2)+(1**2)+(2**2)),1./np.sqrt((1.5**2)+(1**2)+(2*
print(f'The fiber direction N = {N_OE}')

EN_OE = N_OE@Ev@N_OE                     # Normal strain of fiber O->E
print(f'Normal strain on fiber N  = {EN_OE:.4f}')

# normal strain at point E along CF direction:
N_CF = np.array([0, 0, 2.])                        # fiber along E->F
print(f'The fiber direction M = {N_CF}')
EN_CF = N_CF@Ev@N_CF                      #Normal strain of fiber E->F
print(f'Normal strain on fiber M = {EN_CF:.4f}')
```

```
The fiber direction N = [0.5571 0.3714 0.7428]
Normal strain on fiber N  = 0.0000
The fiber direction M = [0. 0. 2.]
Normal strain on fiber M = -0.0160
```

5.

In [ ]:
```python
N_EG = np.array([1.5/np.sqrt((1.5**2)+(1**2)),1./np.sqrt((1.5**2)+(1**2)),0]) # fib

N_EB = np.array([0,0,2]) # fiber along E->B

E_OP = N_EG@Ev@N_EB           #Shear strain between fiber E->G and E->B
print(f'Shear strain between two fibers = {E_OP:.4f}')
print(f'Engineering shear strain between two fibers = {2.*E_OP:.4f}')
```

```
Shear strain between two fibers = -0.0044
Engineering shear strain between two fibers = -0.0089
```

6.

In [ ]:
```python
Ev2 = np.array([[0.005333,0.005333,-0.000667],[0.005333,0.004000,-0.003000],[-0.000
eigenValues, eigenVectors = gr.M_eigen(Ev2)
print('Principal straines (Eigenvalues) =',eigenValues)
```

```
Principal straines (Eigenvalues) = [ 0.0105 -0.     -0.0051]
```

7.

In [ ]:
```python
about, theta = 'y', 30.                    # rotation angle and axis
Ty, about = gr.transferM(theta, about = about)
print(f'Transformation tensor {theta:3.2f}°, w.r.t. {about}:\n{Ty}')
```

```
Transformation tensor 30.00°, w.r.t. y:
[[ 0.866  0.    -0.5  ]
 [ 0.     1.     0.   ]
 [ 0.5    0.     0.866]]
```

In [ ]:
```
T = gr.transferMs()
TU = U.subs(list(zip(T, Matrix(Ty))))
gr.printM(TU, 'Displacement formulas in the new coordinates',n_dgt=4)
```

```
Displacement formulas in the new coordinates
```

Out[ ]:

$$\begin{bmatrix} 0.002667xyz - 0.004 \\ 0.001333xyz - 0.002 \\ -0.002667xyz + 0.004 \end{bmatrix}$$

In [ ]:
```
dE30 = Matrix([TU[0].subs(xE),TU[1].subs(xE),TU[2].subs(xE)])
gr.printM(dE30, 'Displacements in the new coordinates',n_dgt=4)
```

```
Displacements in the new coordinates
```

Out[ ]:

$$\begin{bmatrix} 0.004 \\ 0.002 \\ -0.004 \end{bmatrix}$$

8.

In [ ]:
```
E = Matrix([["\u03B5_11", "\u03B5_12", "\u03B5_13"],
            ["\u03B5_12", "\u03B5_22", "\u03B5_23"],
            ["\u03B5_13", "\u03B5_23", "\u03B5_33"]])

ES = gr.Tensor2_transfer(T,E)  # use the same function used for stress
ES = Matrix(ES)
```

In [ ]:
```
TES = ES.subs(list(zip(T, Matrix(Ty))))
gr.printM(TES, 'Stress formulas in the new coordinates',n_dgt=4)
```

```
Stress formulas in the new coordinates
```

Out[ ]:

$$\begin{bmatrix} 0.75\varepsilon_{11} - 0.866\varepsilon_{13} + 0.25\varepsilon_{33} & 0.866\varepsilon_{12} - 0.5\varepsilon_{23} & 0.433\varepsilon_{11} + 0.5\varepsilon_{13} - 0.433\varepsilon_{33} \\ 0.866\varepsilon_{12} - 0.5\varepsilon_{23} & 1.0\varepsilon_{22} & 0.5\varepsilon_{12} + 0.866\varepsilon_{23} \\ 0.433\varepsilon_{11} + 0.5\varepsilon_{13} - 0.433\varepsilon_{33} & 0.5\varepsilon_{12} + 0.866\varepsilon_{23} & 0.25\varepsilon_{11} + 0.866\varepsilon_{13} + 0.75\varepsilon_{33} \end{bmatrix}$$

In [ ]:
```
TTSv=TES.subs(list(zip(E,Matrix(Ev2))))
gr.printM(TTSv, 'Strain values in the new coordinates', n_dgt=4)
```

```
Strain values in the new coordinates
```

Out[ ]:

$$\begin{bmatrix} 0.003577 & 0.006119 & 0.003708 \\ 0.006119 & 0.004 & 6.842 \cdot 10^{-5} \\ 0.003708 & 6.842 \cdot 10^{-5} & -0.002244 \end{bmatrix}$$