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
```

Homeworks for SM-I course

Homework 2: Understanding stresses and coordinate transformation for stresses

Question 1:

Consider a 2D stress state at a point in a solid, denoted as state A, with stress components given as: σ_{xx} = 88 MPa, σ_{yy}= -35 MPa, σ_{xy} =38 MPa, in the Cartesian coordinate system (x, y).

1. Determine the stress vector on a plane with a normal vector $5\mathbf{i}_1 + 8\mathbf{i}_2$.
2. A new coordinate system (x', y') is created by rigidly rotating (x, y) by 30°, counter-clock wise. Find the stress components in the new coordinate system, denoted as state B.
3. Determine the principal stresses and arrange them in order, using stresses in both (x, y) and (x', y') coordinate systems.
4. Determine the directions of the principal stresses using stresses in both (x, y) and (x', y') coordinate systems.
5. Determine the maximum shear stress, in both (x, y) and (x', y') coordinate systems.
6. Determine the stress invariants, in both (x, y) and (x', y') coordinate systems.
7. Draw the Mohr circle, and put states A and B on the circle.
8. Discuss about the results obtained.

1.

In [ ]:
```
St = np.array([[88,38],[38,-35]])
Normal = np.array([5,8])

S_N, S_NN, S_NS = gr.stressOnN(St,Normal)

print("The stress vector on a plane with a normal vector "+str(Normal)+" is")
print(S_N)
```

```
The stress vector on a plane with a normal vector [5 8] is
[78.8638 -9.54   ]
```

2.

In [ ]:
```
theta = 30

T = np.array([[np.cos(np.radians(theta)),np.sin(np.radians(theta))],[-1*np.sin(np.r

St_30 = gr.Tensor2_transfer(T,St)

print("The new stress components are")
print(St_30)
```

```
The new stress components are
[[ 90.159  -34.2606]
 [-34.2606 -37.159 ]]
```

3.

In [ ]:
```
eigenvalues0, eigenvectors0 = lg.eig(St)

#Sort in order
idx = eigenvalues0.argsort()[::-1]
eigenvalues0 = eigenvalues0[idx]
eigenvectors0 = eigenvectors0[:,idx]
print('Pricipal stress for original coordinate system (Eigenvalues):\n',eigenvalues

eigenvalues30, eigenvectors30 = lg.eig(St_30)

#Sort in order
idx = eigenvalues30.argsort()[::-1]
eigenvalues30 = eigenvalues30[idx]
eigenvectors30 = eigenvectors30[:,idx]
print('Pricipal stress for rotated coordinate system (Eigenvalues):\n',eigenvalues3
```

```
Pricipal stress for original coordinate system (Eigenvalues):
 [ 98.7928 -45.7928]

Pricipal stress for rotated coordinate system (Eigenvalues):
 [ 98.7928 -45.7928]
```

4.

In [ ]:
```
print(f'Principal stress directions for original coordinate system:\n{eigenvectors0

print(f'Principal stress directions for rotated coordinate system:\n{eigenvectors30
```

```
Principal stress directions for original coordinate system:
[[ 0.962  -0.2732]
 [ 0.2732  0.962 ]]

Principal stress directions for rotated coordinate system:
[[ 0.9697  0.2444]
 [-0.2444  0.9697]]
```

5.

In [ ]:
```
maxshear0 = (np.sqrt((4*(St[0,1]**2))+((St[0,0]-St[1,1])**2)))/2

maxshear30 = (np.sqrt((4*(St_30[0,1]**2))+((St_30[0,0]-St_30[1,1])**2)))/2

print("The maximum shear stress for the original coordinate system is %3.2f" % (max
print("The maximum shear stress for the rotated coordinate system is %3.2f" % (maxs
```

```
The maximum shear stress for the original coordinate system is 72.29
The maximum shear stress for the rotated coordinate system is 72.29
```

6.

In [ ]:
```
I1_0 = St[0,0]+St[1,1]
I2_0 = (St[0,0]*St[1,1])-(St[0,1]**2)

print("The stress invariants for the original coordinate system are %3.2f and %3.2f

I1_30 = St_30[0,0]+St_30[1,1]
I2_30 = (St_30[0,0]*St_30[1,1])-(St_30[0,1]**2)

print("The stress invariants for the rotated coordinate system are %3.2f and %3.2f"
```

```
The stress invariants for the original coordinate system are 53.00 and -4524.00
The stress invariants for the rotated coordinate system are 53.00 and -4524.00
```

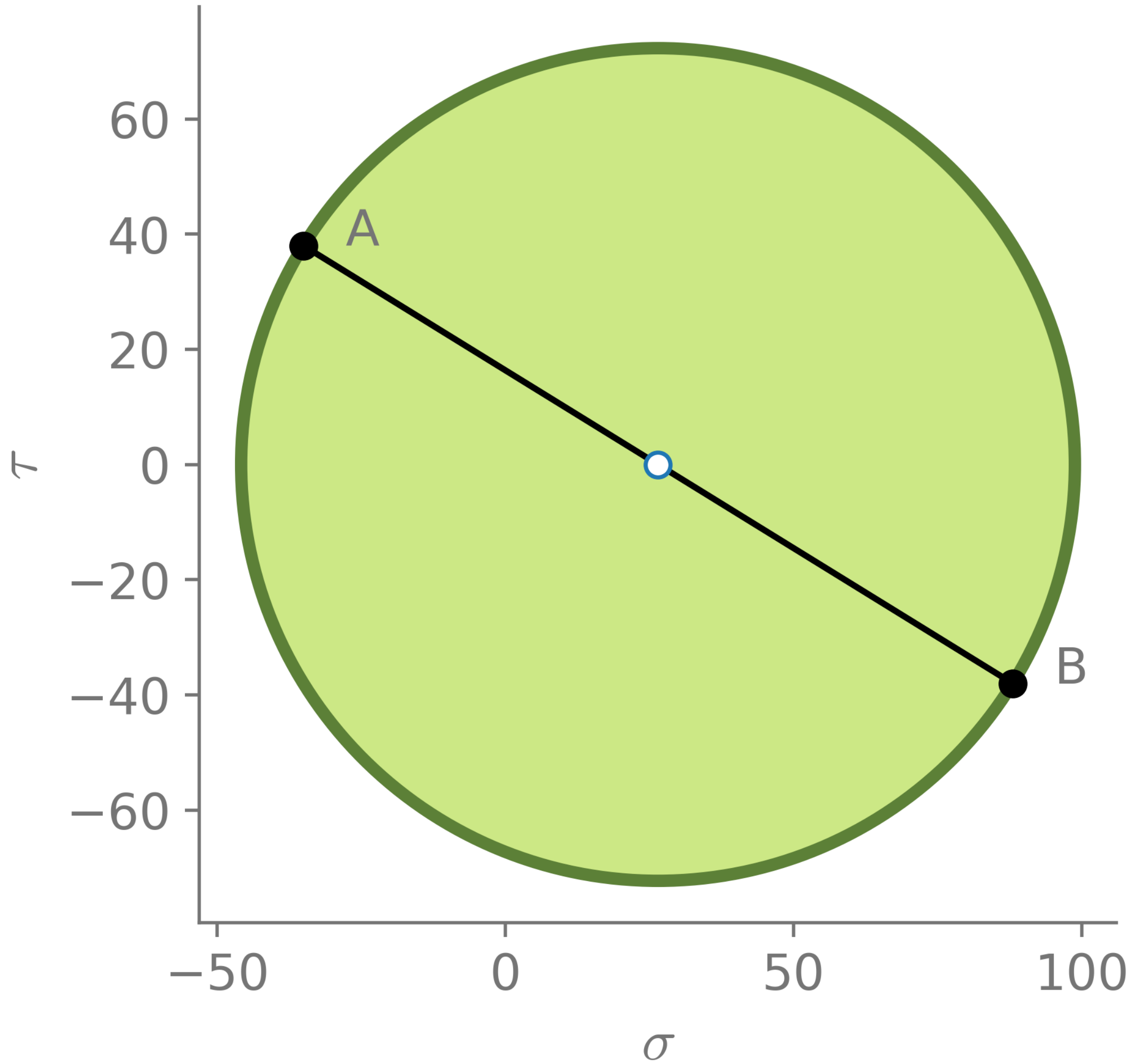
7.

In [ ]:
```
from continuum_mechanics.visualization import mohr2d, mohr3d
from continuum_mechanics.visualization import traction_circle
```

In [ ]:
```
print("Mohr Circle for state A")
mohr2d(St)
```

```
Mohr Circle for state A
```

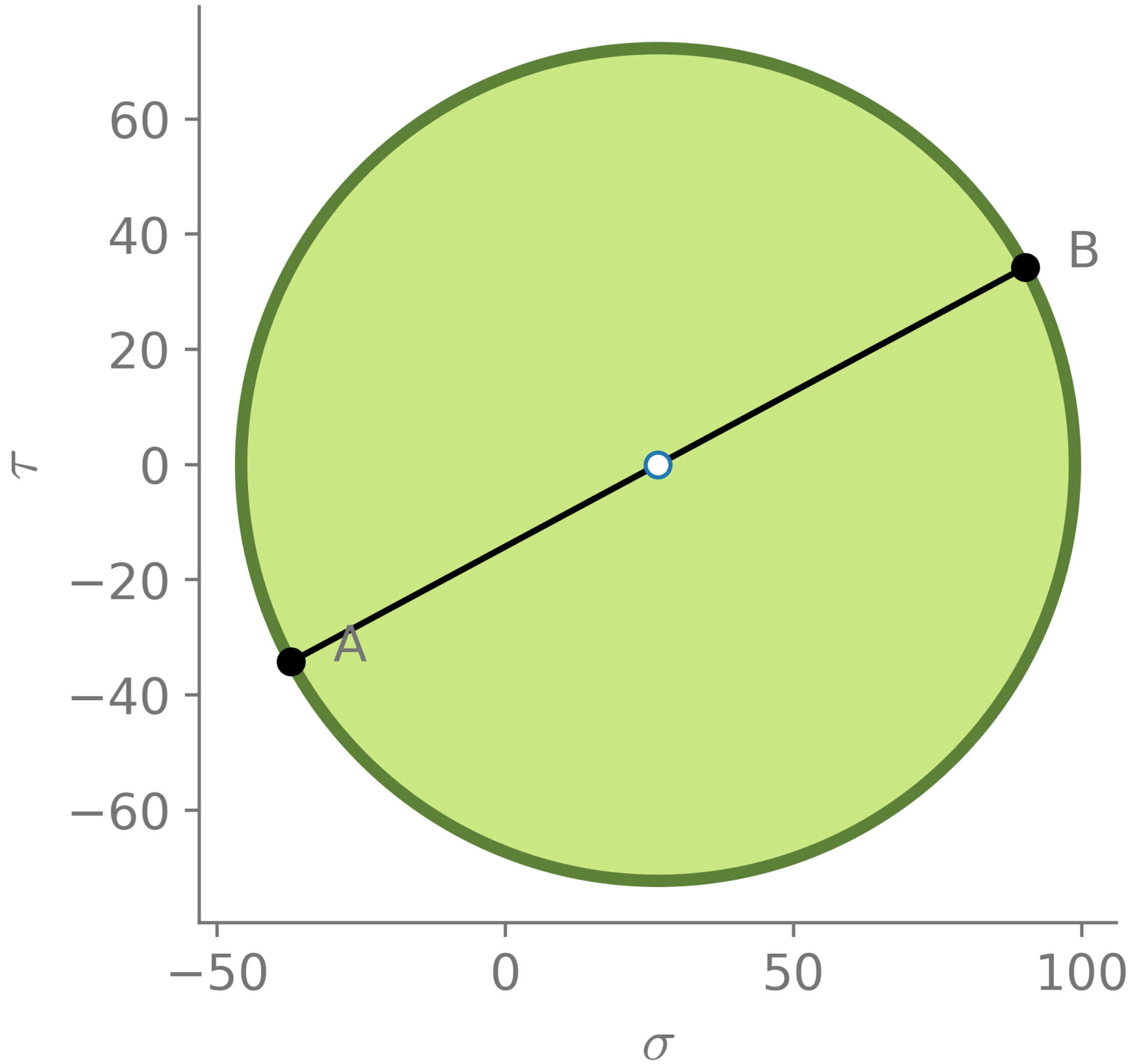
Out[ ]:
```
<Axes: xlabel='$\\sigma$', ylabel='$\\tau$'>
```



In [ ]:
```
print("Mohr Circle for state B")
mohr2d(St_30)
```

```
Mohr Circle for state B
```

Out[ ]: <Axes: xlabel='$\\sigma$', ylabel='$\\tau$'>



8.

The line represents the zero shear stress, which makes sense as the line is rotated 30 degrees with the coordinate transformation.

Question 2:

Consider a 2D stress state at a point in a solid, denoted as state A, with stress components given as: σ_{xx} = 0.5, σ_{yy}= 0.5 , σ_{xy} = -0.5, in the Cartesian coordinate system (x, y).

1. Determine the principal stresses and their directions with respect to (x, y) coordinate system.
2. Determine the maximum shear stress and its direction with respect to (x, y) coordinate system.
3. Draw the Mohr circle.
4. Discuss about the results obtained in relation to the example given in Section 3.7.4.1.

1.

In [ ]:
```
St = np.array([[0.5,-0.5],[-0.5,0.5]])

eigenvalues, eigenvectors = lg.eig(St)

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

print(f'Principal stress directions:\n{eigenvectors}\n')
```

```
Pricipal stress (Eigenvalues):
 [1. 0.]

Principal stress directions:
[[ 0.7071  0.7071]
 [-0.7071  0.7071]]
```

2.

In [ ]:
```
maxshear = (np.sqrt((4*(St[0,1]**2))+((St[0,0]-St[1,1])**2)))/2

print("The maximum shear stress is %3.2f" % (maxshear))
```

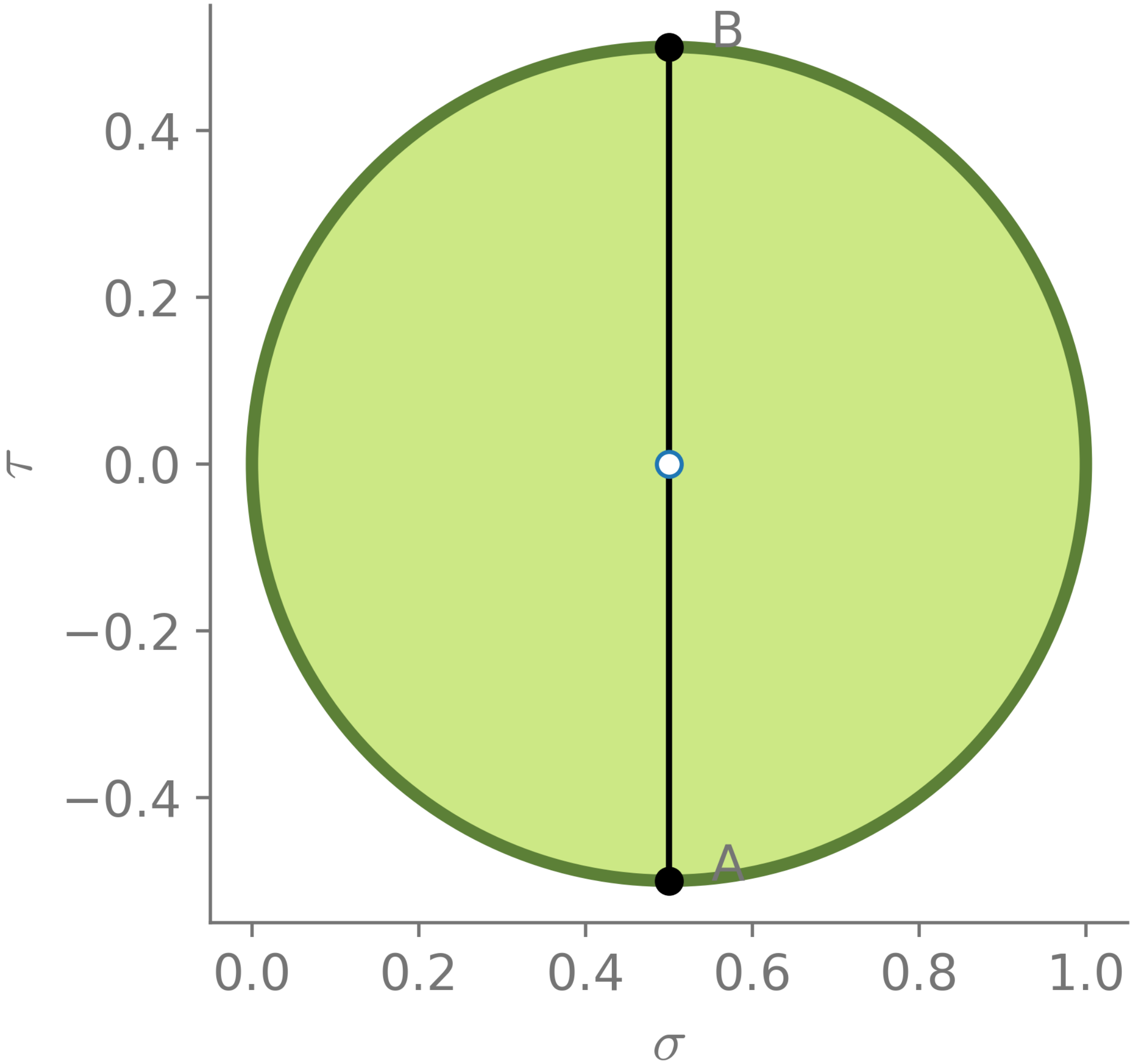
```
The maximum shear stress is 0.50
```

3.

In [ ]:
```
print("Mohr Circle")
mohr2d(St)
```

```
Mohr Circle
```

Out[ ]: <Axes: xlabel='$\\sigma$', ylabel='$\\tau$'>



4.

Similar to the example, This has the the shear stress components in 45 degree rotated coordinates with respect to the Z axis. This means that this material needs to be strongest with these forces, which is consistent with the example

Question 3:

Consider a 3D stress state at a point in a solid with stress components given as: σ_{xx} = 88 MPa, σ_{yy}=-35 MPa, σ_{zz} =28 MPa, σ_{xy} = 12 MPa, σ_{xz}= 28 MPa, and σ_{yz}= 15 MPa.

1. Determine the stress vector on a plane with a normal vector $2\mathbf{i}_1 + 3\mathbf{i}_1 + .5\mathbf{i}_1$.
2. Determine the principal stresses and arrange them in order.
3. Determine the maximum shear stresses.
4. Determine the octahedral shear stress.
5. Determine the von Mises stress.
6. Draw the Mohr circle.

1.

In [ ]:
```
St3d = np.array([[88,12,28],[12,-35,15],[28,15,28]])
Normal = np.array([2,3,0.5])

S_N, S_NN, S_NS = gr.stressOnN(St3d,Normal)

print("The stress vector on a plane with a normal vector "+str(Normal)+" is")
print(S_N)
```

```
The stress vector on a plane with a normal vector [2.  3.  0.5] is
[ 62.087  -20.192   31.5929]
```

2.

In [ ]:
```
eigenvalues, eigenvectors = gr.principalS(St3d)
```

```
Pricipal stress (Eigenvalues):
 [101.0935  18.5646 -38.6581]

Principal stress directions:
[[ 0.9181  0.3933 -0.0485]
 [ 0.1225 -0.1652  0.9786]
 [ 0.3768 -0.9045 -0.1998]]

Possible angles (n1,x)=23.3444338655797∘ or 156.6555661344203∘
```

3.

In [ ]:
```
def principalS_ypr(St):
    '''Diagonalization of a stress tensor in 3D, through cooridinate
    transformation via Yaw, Pitch and Roll.
    Input: St: Stress tensor, array like, 3 by 3
    return: θ, β, γ: Yaw, Pitch and Roll angles, in rad.
            TS_ypr: Diagonalized stress matrix array like, 3 by 3
    '''
    from scipy.optimize import fsolve

    θ, β, γ = symbols("θ, β, γ")  # Angles for  Yaw, Pitch and Roll

    # Create the matrices for Yaw, Pitch, and Roll for transformations:
    Tz = gr.transf_YPRs(θ, about = 'z')[0]
    Ty = gr.transf_YPRs(β, about = 'y')[0]
    Tx = gr.transf_YPRs(γ, about = 'x')[0]
    #print(f'Ty=, {Ty}, \nTx=, {Tx}, \nTz=, {Tz}')

    # Construct the tansformation matrix for Yaw, Pitch and Roll.
    T_ypr = Tz@Ty@Tx

    # Perform coordinate transformation to the given stress tensor.
    S_ypr = T_ypr@St@T_ypr.T

    # Create these three equations:
    eqns = lambda x: [S_ypr[0,1].subs({θ:x[0], β:x[1], γ:x[2]}),
                      S_ypr[0,2].subs({θ:x[0], β:x[1], γ:x[2]}),
                      S_ypr[1,2].subs({θ:x[0], β:x[1], γ:x[2]})]

    # Give an initial guess randomly:
    init_guess = np.random.randn(3)/999.              # make it small

    # Solve the set of three equations numerically:
    sln = fsolve(eqns, init_guess)
    print(f'Solution, θ, β, γ = {sln} (rad)')
    print(f'Solution, θ, β, γ = {sln*180/np.pi} (degree)')

    # Finally, we can check the results, use θ, β, γ  found to
    # perform coordinate trans formation:
    T_ypr_ = T_ypr.subs({θ:sln[0], β:sln[1], γ:sln[2]})
    S_ypr_ = T_ypr_@St@T_ypr_.T

    return θ, β, γ, S_ypr_
```

In [ ]:
```
theta, beta, gamma, S_ypr_ = principalS_ypr(St3d)
```

```
Solution, θ, β, γ = [ 0.0528 -0.4042 -0.1806] (rad)
Solution, θ, β, γ = [  3.0264 -23.1583 -10.3496] (degree)
```

4.

In [ ]:
```
sigma_oct, tau_oct = gr.Oct_stress(eigenvalues)

print("The Octahedral shear stress:")
print(sigma_oct)
```

```
The Octahedral shear stress:
27.0
```

5.

```
In [ ]: vonMises = gr.von_Mises(eigenvalues)

        print("The von Mises stress:")
        print(vonMises)
```

```
The von Mises stress:
121.68812596141007
```

6.

```
In [ ]: mohr3d(St3d)
```

```
Out[ ]: <Axes: xlabel='$\\sigma$', ylabel='$\\tau$'>
```