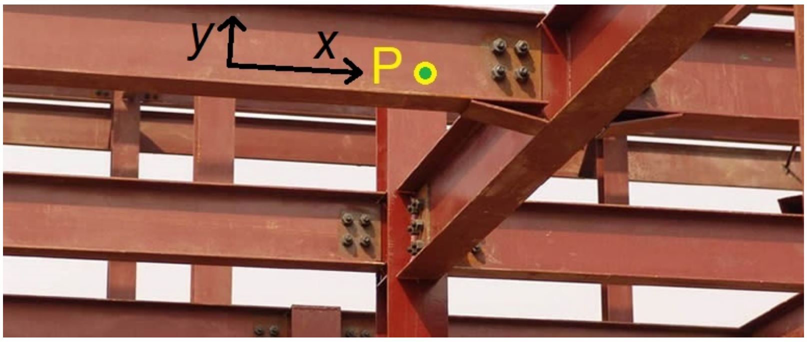
1 Példa 5.3

Egy acélszerkezetet alkotó I-szelvény P pontjában nyúlásmérő bélyeget ragasztunk és mérjük az alakváltozást a, b, c irányokban. Határozzuk meg a feszültségi állapotot.



In [1]:

```
import sympy as sp
σ_x,σ_y,τ_xy,ε_x,ε_y,ε_z,γ_xy = sp.symbols('σ_x,σ_y,τ_xy,ε_x,ε_y,ε_z,γ_xy')

ε_a = 100e-6         # -
ε_b = 50e-6          # -
ε_c = -70e-6         # -
α_a = 0              # rad
α_b = 70/180*sp.pi   # rad
α_c = 200/180*sp.pi  # rad
E   = 210e3          # MPa
ν   = 0.3            # -
```

executed in 354ms, finished 17:34:35 2020-04-16

In [2]:

```
# Definiáljunk 3 db. 'sp.Matrix'-ot az irányvektorok megadására!
# A szögletes zárójelekre azért van szükség, mert a 'Matrix' egy sima python 'l
# kér bemenetül, amit aztán a saját adattípusára alakít át

n_a = sp.Matrix([sp.cos(α_a), sp.sin(α_a), 0])
n_b = sp.Matrix([sp.cos(α_b), sp.sin(α_b), 0])
n_c = sp.Matrix([sp.cos(α_c), sp.sin(α_c), 0])

display(n_a.evalf(5))
display(n_b.evalf(5))
display(n_c.evalf(5))
```

executed in 36ms, finished 17:34:35 2020-04-16

$$\begin{bmatrix} 1.0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0.34202 \\ 0.93969 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -0.93969 \\ -0.34202 \\ 0 \end{bmatrix}$$

In [3]:

```
# Definiáljuk a feszültség és alakváltozás tenzorokat a nyúlás irányokhoz hason

σ = sp.Matrix([
[σ_x, τ_xy, 0],
[τ_xy, σ_y, 0],
[0,     0,   0]])

ε = sp.Matrix([
[ε_x,     γ_xy/2, 0  ],
[γ_xy/2, ε_y,     0  ],
[0,       0,      ε_z]])
```

executed in 3ms, finished 17:34:35 2020-04-16

A bélyeg által mért nyúlások nagyságát kifejezhetjük az alakváltozási tenzor és az irányvektorok kettős skalárszorzatával:

$$\varepsilon_a = \mathbf{n}_a^\top \boldsymbol{\varepsilon} \mathbf{n}_a,$$

$$\varepsilon_b = \mathbf{n}_b^\top \boldsymbol{\varepsilon} \mathbf{n}_b,$$

$$\varepsilon_c = \mathbf{n}_c^\top \boldsymbol{\varepsilon} \mathbf{n}_c.$$

Ez három skaláregyenletet ad, melyekből az alakváltozási tenzor elemei meghatározhatóak a már ismert Sympy solverrel.

In [4]:

```
print("A skalár egyenletek jobb oldalai:")
display((n_a.T*ε*n_a).evalf(5))
display((n_b.T*ε*n_b).evalf(5))
display((n_c.T*ε*n_c).evalf(5))

# A skalárszorzatok eredményeit egy elemű 'list' adatként kapjuk, amiket az ind
# Az egyenletek 0-ra rendezve
eq1 = (n_a.T*ε*n_a)[0]-ε_a
eq2 = (n_b.T*ε*n_b)[0]-ε_b
eq3 = (n_c.T*ε*n_c)[0]-ε_c

# Hogy a solve könnyebben boldoguljon az egyenletmegoldásokkal,
# egyszerűsítsük az egyenletek numerikus részeit az 'evalf'-el
eq1 = eq1.evalf()
eq2 = eq2.evalf()
eq3 = eq3.evalf()

# Mivel a fenti egyenletek lineárisak a változóikban, így a sima 'solve' helyet
# Ez sokkal kedvezőbb futási idővel rendelkezik, és stabilabban működik az álta
sol1 = sp.solve([eq1,eq2,eq3],ε_x,ε_y,γ_xy)

# a numerikus eredmények (a 'dictionary'-ből kiszedve):
ε_xn  = sol1[ε_x]
ε_yn  = sol1[ε_y]
γ_xyn = sol1[γ_xy]

print("A megoldások:")

print("ε_xn = " + str(ε_xn)) # -
print("ε_yn = " + str(ε_yn)) # -
print("γ_xyn = " + str(γ_xyn)) # -
```

executed in 757ms, finished 17:34:36 2020-04-16

```
A skalár egyenletek jobb oldalai:
```

$$\left[\varepsilon_x\right]$$

$$\left[0.32139\gamma_{xy} + 0.11698\varepsilon_x + 0.88302\varepsilon_y\right]$$

$$\left[0.32139\gamma_{xy} + 0.88302\varepsilon_x + 0.11698\varepsilon_y\right]$$

```
A megoldások:
ε_xn = 0.000100000000000000
ε_yn = 0.000256648874719873
γ_xyn = -0.000585961628761869
```

In [5]:

```
sol_new = sp.solve([eq1,eq2,eq3],ε_x,ε_y,γ_xy)
```

executed in 200ms, finished 17:34:36 2020-04-16

A még ismeretlen ε_z értékét a Hooke törvényből számíthatjuk:

In [6]:

```
ε_zn = -ν/(1-ν)*(ε_xn+ε_yn)
display(ε_zn.evalf(5)) # -
```

executed in 6ms, finished 17:34:36 2020-04-16

$$-0.00015285$$

A feszültségi állapot az alakváltozási állapotból meghatározható az általános Hooke törvény segítségével:

$$\sigma = \frac{E}{1+\nu}\left[\varepsilon + \frac{\nu}{1-2\nu}\varepsilon_I \mathbf{I}\right].$$

Ehhez a numerikusan kiszámított alakváltozás komponensekből összeállítunk egy numerikus alakváltozási tenzort.

In [7]:

```
ε_n = sp.Matrix([
[ε_xn,    γ_xyn/2, 0  ],
[γ_xyn/2, ε_yn,    0  ],
[0,       0,        ε_zn]])

print("Az alakváltozási tenzor: ")
display(ε_n.evalf(5))

# Az első skalár invariáns:
ε_I = ε_n[0,0]+ε_n[1,1]+ε_n[2,2]

# Az egységmátrixot a Sympy 'eye' függvényével tudjuk létrehozni, melynek argum
σ_n = E/(1+ν)*(ε_n+ν/(1-2*ν)*ε_I*sp.eye(3))

# Vegyük észre a numerikus számításból adódó hibát!
print("A feszültségi tenzor: ")
display(σ_n.evalf(5)) #MPa
```

executed in 27ms, finished 17:34:36 2020-04-16

```
Az alakváltozási tenzor:
```

$$\begin{bmatrix} 0.0001 & -0.00029298 & 0 \\ -0.00029298 & 0.00025665 & 0 \\ 0 & 0 & -0.00015285 \end{bmatrix}$$

```
A feszültségi tenzor:
```

$$\begin{bmatrix} 40.845 & -47.328 & 0 \\ -47.328 & 66.15 & 0 \\ 0 & 0 & -8.757 \cdot 10^{-15} \end{bmatrix}$$

A 4.9-4.10 feladatokban bemutatott és alkalmazott saját függvénykönyvtár segítségével kiszámíthatjuk a főfeszültségeket és főnyúlásokat.

In [8]:

```
from sajat_fuggvenyeket_tartalmazo_fajl import print_eigensystem_feszultseg
from sajat_fuggvenyeket_tartalmazo_fajl import print_eigensystem_alakvaltozas

print_eigensystem_feszultseg(σ_n)
print_eigensystem_alakvaltozas(ε_n)
```

executed in 600ms, finished 17:34:37 2020-04-16

```
1. Főfeszültség: 102.49 MPa
1. Főirány:
```

$$\begin{bmatrix} -0.60899 \\ 0.79318 \\ 0 \end{bmatrix}$$

```
2. Főfeszültség: 4.5076 MPa
2. Főirány:
```

$$\begin{bmatrix} 0.79318 \\ 0.60899 \\ 0 \end{bmatrix}$$

```
3. Főfeszültség: -8.7570e-15 MPa
3. Főirány:
```

$$\begin{bmatrix} 0 \\ 0 \\ 1.0 \end{bmatrix}$$

```
1. Főnyúlás: 0.00048159
1. Főirány:
```

$$\begin{bmatrix} -0.60899 \\ 0.79318 \\ 0 \end{bmatrix}$$

```
2. Főnyúlás: -0.00012495
2. Főirány:
```

$$\begin{bmatrix} 0.79318 \\ 0.60899 \\ 0 \end{bmatrix}$$

```
3. Főnyúlás: -0.00015285
3. Főirány:
```

$$\begin{bmatrix} 0 \\ 0 \\ 1.0 \end{bmatrix}$$